Traductor de Pictogramas a Texto

Trabajo de Fin de Grado
Curso 2018–2019

Autores
Salvador González Álvarez
José María López Pulido

Directoras
Virginia Francisco Gilmartín
Susana Bautista Blasco

Trabajo de Fin de Grado en Ingeniería del Software
Facultad de Informática
Universidad Complutense de Madrid

Traductor de Pictogramas a Texto

Trabajo de Fin de Grado en Ingeniería del Software

Autores
Salvador González Álvarez
José María López Pulido

Directoras
Virginia Francisco Gilmartín
Susana Bautista Blasco

Convocatoria: *Junio 2019*

Trabajo de Fin de Grado en Ingeniería del Software
Facultad de Informática
Universidad Complutense de Madrid

31 de mayo de 2019

Autorización de difusión

Los abajo firmantes, matriculado en el Grado de Ingeniería del Software de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores el presente Trabajo Fin de Grado: "Traductor de Pictogramas a Texto", realizado durante el curso académico 2018/2019 bajo la dirección de Virginia Francisco Gilmartín y Susana Bautista Blasco en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Salvador González Álvarez José María López Pulido

31 de mayo de 2019

Agradecimientos

En primer lugar, queremos agradecer profundamente a nuestras directoras del proyecto, Virginia y Susana, por su dedicación, tiempo y cercanía con nosotros y con el trabajo. Ha sido un verdadero placer trabajar con dos profesoras tan buenas tanto a nivel profesional como humano. Sus directrices e indicaciones nos han ayudado muchísimo para avanzar en el trabajo y sus conocimientos sobre la materia nos han aportado un valor incalculable de sabiduría ante cualquier obstáculo que nos íbamos encontrando por el camino.

No queremos olvidarnos tampoco de todas aquellas personas que nos han ayudado en alguna etapa de este proyecto, con una mención especial para las personas que forman parte de Arasaac por dejarnos utilizar su API sin ninguna condición y por mostrarse accesibles y cercanos las veces que nos hemos puesto en contacto con ellos. Agradecer a Antonio por la ayuda prestada tanto para el despliegue de Pict2Text, como por la resolución de dudas sobre el procesamiento de lenguaje natural.

Para finalizar, queríamos agradecer a nuestras familias, parejas y amigos, ya que este trabajo es en gran medida reflejo de su incondicional apoyo y animo.

Resumen

La comunicación es fundamental en la interacción social y posibilita la integración y el intercambio. Sin embargo, hay personas con ciertas diversidades funcionales que afectan a su comunicación verbal y deben usar pictogramas como apoyo en la comunicación. No todo el mundo entiende la comunicación mediante pictogramas y esto impide a las personas que necesitan usarlos para comunicarse tener una comunicación efectiva, por más simple que pudiera llegar a ser lo que quieren decir. En este TFG hemos desarrollado una aplicación web para ayudar a aquellas personas con dificultades en la comunicación a integrarse en el medio social, traduciendo textos escritos con pictogramas a texto escrito en lenguaje natural en castellano. En la actualidad, no existe ninguna aplicación que haga una traducción de pictogramas a texto, por lo que nuestro TFG puede ayudar a cubrir esa laguna y ayudar a mejorar la calidad de vida de las personas que necesitan los pictogramas para comunicarse. La aplicación que hemos desarrollado en este TFG ha sido implementada usando un diseño basado en componentes, lo que permite a otros desarrolladores integrar cualquiera de nuestras componentes en sus aplicaciones. Se ha realizado una evaluación de las traducciones realizadas por nuestra aplicación con el fin de comprobar la corrección de estas y los resultados obtenidos son bastante alentadores aunque queda mucho margen de mejora. Podemos afirmar que con la aplicación desarrollada en este TFG, se han establecido unas bases sólidas sobre las que seguir trabajando para mejorar la cobertura y corrección de los textos traducidos.

Palabras clave

- Pictogramas.
- Aplicación Web.
- Generación Lenguaje Natural.
- Servicios web.
- ARAASAC.

Abstract

Communication is fundamental in social interaction and it makes integration and exchange possible. However, there are people who have certain functional diversities that affect their verbal communication so they must use pictograms to support communication. Not everyone understands messages with pictograms and this prevents people who need to use them to communicate to have an effective communication. In this Final Degree Project we have developed a web application to help those people with communication difficulties to integrate into social enviroment by translating texts written with pictograms to texts written in natural language in Spanish. Now days, there is no application that makes a translation of pictograms into text, so our project can help to filling that gap helping to improve the quality of life of people who need pictograms to communicate. The application has been developed using a component-based design, which allows other developers to integrate any of our components into their applications. Finally, we have made an evaluation of the translations made by our application has been carried out to check the correctness of these and the results obtained are quite encouraging although there is still a lot of room for improvement. We can affirm that with the application developed in this work, solid bases have been established on which to continue working to improve the coverage and the correction of translated texts.

Keywords

- Pictogram.
- Web Aplication.
- Natural Language Generation.
- Web services.
- ARAASAC.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se presentan, la motivación que hay detrás de este TFG, los objetivos que nos marcamos al comienzo del TFG y la metodología de desarrollo que hemos seguido, a lo largo del trabajo. Además, al final del capítulo indicamos la estructura que sigue este documento.

1.1. Motivación

La comunicación es una parte fundamental del desarrollo social y humano, pero algunas personas con diversidad funcional encuentran muchos obstáculos para conseguir una comunicación efectiva. La aparición de diferentes vías de comunicación, como los pictogramas (imágenes gráficas que se relacionan con algún objeto o concepto) han supuesto un gran avance para la comunicación en este colectivo. Pero la realidad es que la comunicación sigue estando muy limitada y en ocasiones se reduce a personas con el mismo tipo de diversidad funcional y especialistas formados en este área. Este es el caso de las personas cuya vía de comunicación son los pictogramas, ya que muchas personas desconoce el significado de los pictogramas, lo que hace que la comunicación con este tipo de personas quede notablemente reducida o en muchos casos sea imposible.

Por todo ello, existe la necesidad de crear una herramienta que facilite la comunicación de aquellas personas que emplean pictogramas para comunicarse con su entorno, un software que sea capaz de traducir pictogramas a lenguaje natural. Dado que actualmente existen múltiples herramientas que permiten la traducción de lenguaje natural a pictogramas pero no de pictogramas a lenguaje natural. Esta herramienta software permitirá derribar barreras y ayudar a la inclusión social de personas que por su diversidad funcional están en riesgo de exclusión.

Los principales beneficiados de esta herramienta serán usuarios con pro-

blemas para comunicarse mediante el lenguaje natural y que necesitan apoyarse en los pictogramas, como pueden ser personas con parálisis cerebral, autismo o cualquier otro tipo de discapacidad cognitiva. Además, este trabajo está pensado para ayudar a esas personas a poder comunicarse en cualquier situación cotidiana como puede ser pedir un refresco en un bar, expresar su deseo de comer macarrones, o expresar aquellas actividades que han realizado o van a realizar en un futuro como puede ser ir a una excursión. Algo tan simple que, por desgracia, para ellos no lo es tanto.

1.2. Objetivos

El objetivo principal de este trabajo es la creación de un traductor de textos escritos con pictogramas a texto escrito a lenguaje natural en castellano. El traductor debe generar un texto gramaticalmente correcto y no limitarse a traducir cada pictograma con la palabra que representa.

A la hora de diseñar la aplicación se buscará su facilidad de uso a través de una interfaz cómoda, accesible e intuitiva, para de esta manera llegar al máximo publico posible.

Además, para implementar cada una de las funcionalidades de nuestra aplicación, usaremos servicios web con el fin de poder crear diferentes módulos independientes que se pueden integrar en otras aplicaciones. Esto sera muy útil para otros programadores que estén haciendo aplicaciones similares ya que podrían integrar aquellas funcionalidades desarrolladas en nuestro trabajo que les interesen.

Con este TFG también buscamos afianzar los conocimientos adquiridos en el Grado de Ingeniería de Software que hemos cursado en los últimos años. Además de adquirir nuevas competencias antes de salir a la vida laboral como graduados universitarios.

1.3. Metodología de Trabajo

Durante este TFG seguiremos una metodología con algunas de las características típicas de las metodologías ágiles. Necesitamos una metodología adaptativa, dado que al comienzo del TFG se desconoce la viabilidad del proyecto, el alcance y el como desarrollarlo.

La comunicación con los tutores y entre nosotros, sera constante desde el principio del proyecto con el fin de garantizar la calidad del producto y obtener asesoramiento. Estaremos en contacto a través del correo para resolver dudas y, además cada dos o tres semanas tendrá lugar una reunión para revisar el trabajo realizado y para establecer las prioridades del trabajo restante y seleccionar lo que se hará hasta la próxima reunión.

Para intentar asegurar la correcta resolución del proyecto se realizaran

entregas constantes tanto de memoria como de código. Dos o tres veces al mes la memoria y una versión funcional de la aplicación serán revisadas por nuestras tutoras, comprobando así si vamos o no por el buen camino.

Se utilizará un tablero Kanban para representar el estado del proyecto en cada momento. Esto además ayudara a organizarnos, ya que existen limitaciones como no trabajar en el mismo sitio, ni con el mismo horario y necesitamos un tablero que nos permitiera de una manera rápida saber el estado del trabajo en cada momento. La prioridad del trabajo a realizar será siempre en base al valor que otorga esa tarea al producto final. El tablero Kanban será un radiador de información para todo el equipo, ya que de un vistazo cada persona podrá conocer el estado del proyecto y de cada una de las tareas. Las columnas del tablero son:

- To Do: En esta columna aparecen listadas aquellas tareas que están por realizar. Las tareas están ordenadas por prioridad de manera que arriba siempre aparecen las más prioritarias y que pasaran a ser desarrolladas. Después de cada reunión se añadirán nuevas tareas y se reajustaran las prioridades.

- In Progress: Tareas que se están desarrollando. Cada tarea en esta columna debe estar asignada a uno de los miembros del equipo de trabajo y una misma persona no puede tener más de una tarea asignada en esta columna, por tanto el WIP(Work In Progress)[1], de esta columna en nuestro trabajo es dos.

- Test: Todas las tareas finalizadas por un miembro del equipo pasan a está columna para ser revisadas por el otro.

- Revisión: Tareas que las tutoras están revisando, para validar su corrección antes de la siguiente reunión.

- Done: Tareas que tras el proceso de revisión se pueden dar por terminadas.

Se trabajará por sprints de dos o tres semanas. Al final de nuestras reuniones con nuestras tutoras entre todos estableceremos la duración del siguiente sprint y cuales serán las tareas a realizar en el mismo.

El sistema elegido para el control de versiones es GitHub, utilizando la cuenta del grupo NIL[2]. Utilizar esta cuenta nos permitirá tener el proyecto en privado y que solo los miembros del mismo tengan acceso. Una vez finalizado este TFG haremos pública la cuenta para que cualquiera pueda consultar el proyecto.

[1] https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-limite-wip/
[2] https://github.com/NILGroup/TFG-1819-PictoTexto

1.4. Estructura de la memoria

La memoria de este trabajo estará estructurada de la siguiente manera. En el Capítulo 2 **(Estado de la cuestión)**, hablaremos de todos las aspectos importantes para la Generación de Lenguaje Natural, de los Sistemas Aumentativos y Alternativos de Comunicación, y más concretamente los pictogramas, y de los diferentes sistemas pictográficos existentes en la actualidad, además se introducirán los servicios web. En el Capítulo 3 **(Herramientas)** será donde se describan las diferentes herramientas utilizadas para la confección de este trabajo. Aquí se presentarán Django, Angular, Simple-NLG, la API de Arasaac usada para obtener la información necesaria sobre los pictogramas y el analizador sintáctico Spacy. En el Capítulo 4 **(Pict2Text)**, se presenta en detalle la aplicación desarrollada en este TFG. En el Capítulo 5 **(Evaluación)** se presentaran los resultados de la evaluación realizada. En el Capítulo 6 **(Trabajo individual)** se expondrá el trabajo individual realizado por cada uno de los autores del TFG. Por ultimo en el Capítulo 7 **(Conclusiones y Trabajo Futuro)** expondremos las conclusiones que hemos sacado al finalizar este TFG, y cuales serían los siguientes pasos para la mejora de la aplicación.

Chapter 1

Introduction

In this chapter will be presented, the motivation behind this project, the goals that were set at the beginning of the project and the methodology that we have followed throughout the work. Also, at the end of this chapter we present the structure of this document.

1.1. Motivation

Communication is a fundamental part of social and human growth, but some people with functional diversity find many handicaps to achieve effective communication. The appearance of different communication channels, such as pictograms(graphic images that represent a object or a concept), has been a great advance for communication in this collective. But the reality is that communication is still very limited and sometimes it is limited to people with the same type of functional diversity or specialists trained in this area. This is the case of people whose way of communication are pictograms, many people do not know the meaning of the pictograms, which makes the communication with this people impossible or notably reduced.

For that reasons there is a need to create a tool that facilitates communication of people who use pictograms to comunicate, a software capable of translating pictograms into natural language in spanish. There are currently multiple tools that allow the translation of natural language to pictograms but not tools which translate pictograms into natural language. This software tool will allow to break down barriers and help the inclusion of people with functional disabilities who are at risk of exclusion.

The main beneficiaries of this tool will be people with communication problems who need to rely on the pictograms to communicate, such as people with cerebral palsy, autism or any other type of cognitive disability. Also, this work is designed to help these people to be able to communicate in any

daily situation such as being able to order a refreshment in a bar, expressing their desire to eat macaroni, or expressing activities they have done or will be doing in the future, such as go on a excursion. Something so simple that unfortunately for these people it is not so much.

1.2. Goals

The main goal of this work is the creation of translator os text written with pictograms to texts written in natural language. The translator must create a grammatically correct text in Spanish and not just translate each pictogram by the word that represents.

The application, should be easy to use and should have a comfortable, accessible and intuitive interface, to reach the maximum possible public.

Also, to develop each of the functionalities of our application, we will use web services to create different and independent modules that can be integrated into other applications. This will be very useful for other developers who can integrate our services in their application if they wish.

In this work we also seek to consolidate the knowledge acquired during the Software Engineering Degree and acquire new skills before leaving to work as university graduates.

1.3. Work Methodology

During this project we will follow a methodology with some of the typical characteristics of agile methodologies. We need an adaptive methodology given that at the beginning of the project we unknown if the project was viable, or how to develop it.

The communication with the tutors and between us will be constant since the beginning of the project to guarantee the quality of the product and obtain advice. We will be in contact through the mail to answer questions and, in addition, every two or three weeks there will be a meeting to review the work done and to establish the priorities of the remaining work and select what would be done until the next meeting.

To make secure that the project is on track at all times there will be constant deliveries of both memory and code. Two o three times at month, the memory and a functional version of the aplication will be revised by our advisors, to verify that we and to right track.

We will use a Kanban board to represent the status of the project in every moment. This board will also helped us with the organization, because there were limitations such as not working in the same place, or with the same schedule and we needed a board to obtain a glance status of the project. The priority of the work to be done will always be based on the value that these

tasks granted to the final product. The Kanban board will be an information radiator for all the team, because at a glance each person could know the status of the project and each of it´s tasks. The columns of the board are:

- To Do: In this column are listed those tasks that are about to do. The tasks are ordered by priority so that at the beginning always appear the most priority and develop. After each meeting we added new tasks and readjusted the priorities.
- In Progress: Tasks that are being developed. Each task in this column must be assigned to one of the members of the work team and the same person can not have more than one task assigned in this column, so the WIP(Work In Progress)[1] of this column in our work is two.
- Test: All tasks completed by a team member are moved to this column to be reviewed by the other member of the team.
- Revision: Tasks that the tutors are reviewing, to validate their correction before the next meeting.
- Done: Tasks that after the review process are classified as done.

We have worked through sprints of two or three weeks. At the end of our meetings with our advisors we established the duration of the next sprint as well as the tasks to do during it.

The version control system chosen is GitHub, using the NIL group account[2]. Using this account allowed us to have a private project that only members of the project could access. Once this project was finalized we will make the account public so that everyone could consult it.

1.4. Document structure

This document is structured as follows. In Chapter 2 **(Estado de la cuestión)** we will talk about all the important aspects for the Generation of Natural Language, of the Augmentative and Alternative Communication Systems, and more specifically the pictograms, and the different pictographic systems currently available, and also we will introduce web services. In Chapter 3 **(Herramientas)** the different tools used for the preparation of this work are described. In this chapter we will present Django, Angular, SimpleNLG, the Arasaac API used to obtain the necessary information about the pictograms, and Spacy. In Chapter 4 **(Pict2Text)**, the application developed in this project will be presented in detail. In Chapter 5 **(Evaluación)**

[1] https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-limite-wip/
[2] https://github.com/NILGroup/TFG-1819-PictoTexto

the results of the evaluation carried out will be presented. In Chapter 6 **(Trabajo individual)** the individual work carried out by each of the authors of this work will be exposed. Finally in Chapter 7 **(Conclusiones y Trabajo Futuro)** we will present the conclusions and the future work.

Capítulo 2

Estado de la Cuestión

Como ya se ha explicado en el Capítulo 1, nuestro objetivo es crear una aplicación que permita la traducción de un texto escrito con pictogramas a lenguaje natural. El primer paso para conseguir esto sera entender que son los Sistemas Aumentativos y Alternativos de Comunicación (SAAC) y los diferentes tipos (sección 2.1). A continuación, en la sección 2.2, profundizaremos en los pictogramas y en como es la comunicación con ellos. Una vez sentadas las bases de la comunicación con pictogramas, en la sección 2.3 daremos un breve repaso a los servicios de traducción de texto a pictogramas existentes. A continuación, en la sección 2.4 se explicará que es la Generación de Lenguaje Natural (GLN), dado que en nuestro trabajo sera necesaria para generar el texto traducido. Por ultimo en la sección 2.5, hablaremos brevemente sobre los servicios web, exponiendo cuales son sus ventajas y desventajas.

2.1. Sistemas Aumentativos y Alternativos de Comunicación

La comunicación es una necesidad, pero en casos de autismo o de parálisis cerebral la comunicación esta limitada porque el uso del lenguaje verbal es un reto. En estos casos se necesitan sistemas de comunicación de lenguaje no verbales que sustituyan o sirvan de apoyo al lenguaje verbal durante la comunicación.

Los Sistemas Aumentativos y Alternativos de Comunicación[1] (SAAC) son distintas formas de expresar el lenguaje natural, que tienen como objetivo aumentar y/o compensar los problemas de comunicación de muchas personas con dificultades para conseguir una expresión verbal funcional. El término Comunicación Aumentativa describe las formas que usa el individuo para

[1] `http://www.arasaac.org/aac.php`

comunicarse, cuando el uso del lenguaje natural no es suficiente, mientras que el termino Comunicación Alternativa hace referencia a diferentes métodos o vías de comunicación que sustituyen completamente el lenguaje oral, como por ejemplo los pictogramas o el lenguaje de signos (Correa Piñero et al., 2011).

Lo que se pretende con los SAAC es conseguir una comunicación funcional, adecuada y generalizable, que permita al individuo expresarse y alcanzar una mayor integración en su entorno social. Los SAAC ayudan a mejorar la calidad de vida de las personas con problemas para comunicarse, al otorgar al individuo una vía para mejorar su comunicación. Cabe destacar que los SAAC no son usados exclusivamente por personas con alteraciones en la comunicación, sino que son utilizados por todos a diario, por ejemplo cuando empleamos gestos, señales de trafico, etc.

Los SAAC se pueden clasificar en dos grandes grupos (Romski y Sevcik, 1997; Basil et al., 1990; Uribe et al., 2018):

1. SAAC sin ayuda: Sistemas que carecen de soporte físico. Abarca desde el uso de mímica y gestos hasta el uso de signos manuales.

2. SAAC con ayuda: sistemas que necesitan un soporte físico independiente del emisor. Este tipo abarca desde sistemas sencillos basados en dibujos o imágenes hasta objetos reales en miniatura o los sistemas de comunicación escritos como el Braile. Están orientados a personas con problemas en el habla o con dificultades cognitivas o de aprendizaje. Dentro de los SAAC con ayuda se encuentran los pictogramas de los cuales hablaremos en la sección 2.2.

Se ha demostrado que los SAAC son indispensables para aquellas personas que en mayor o menor medida, tienen impedido el uso del lenguaje. Sus usuarios potenciales son aquellas personas que por patologías como el autismo, la parálisis cerebral, o lesiones cerebrales, están impedidas para expresarse de manera adecuada o correcta (Warrick, 2010). Además de ser un recurso necesario para estas personas se ha demostrado que tienen múltiples beneficios siendo los principales (Álvarez, 2013):

1. Ayudan al desarrollo de la comunicación, proporcionando estrategias al usuario.

2. Posibilitan el desarrollo personal y social ademas de fomentar las relaciones interpersonales.

3. Evitan el aislamiento, fomentando habilidades afectivo-sociales.

Cabe destacar que la complejidad de los SAAC depende del nivel cognitivo del usuario (Abril Abadín et al., 2010). Cada persona es diferente, y por ejemplo en el uso de imágenes, el nivel de abstracción y complejidad debe adecuarse a las necesidades de cada individuo.

2.2. Pictogramas

Se define pictograma como un signo icónico dibujado y no lingüístico, que representa un objeto real o un significado (Correa Piñero et al., 2011). Se engloban dentro de los SAACS con ayuda, y se aplican a personas que no están alfabetizadas ya sea por causa de la edad o de alguna discapacidad. En la Figura 2.1 podemos observar un ejemplo de pictograma que representa un smartwatch.

La utilización de pictogramas para la escritura o pictografía se remonta al neolítico, donde se utilizaban piedras talladas para la representación de signos de cierta semejanza con el objeto que representan. El origen de los sistemas pictográficos actuales se sitúa en 1981 con Mayer-Johnson[2] y se caracteriza por una facilidad de interpretación.

Figura 2.1: Pictograma que representa un smarthwatch.

Las características principales de los pictogramas son las siguientes[3]:

1. Guardan relación con aquello que representan, tal y como se puede ver en el pictograma de la Figura 2.1 el cual representa un smarthwach.

2. Son elementos gráficos que combinados representan el objeto tomado como referente. Por ejemplo en el pictograma de la Figura 2.2 la cara, el lápiz de labios y la mano se juntan en un solo pictograma para representar la acción de pintarse los labios.

3. La imagen debe de ser comprensible por el mayor número de personas, independientemente de la formación, idioma o discapacidad.

4. A la hora de construir un pictograma, se deben de seguir unas reglas que le permitan mantener una coherencia visual, es decir la legibilidad de un pictograma debe de ser inmediata.

5. Deben de ser sencillos y representar únicamente los elementos más importantes, evitando posibles estímulos distractores o información irre-

[2] https://www.uv.es/bellochc/logopedia/NRTLogo8.wiki?8
[3] https://smileandlearn.com/que-es-un-pictograma/

Figura 2.2: Pictograma que representa pintarse los labios.

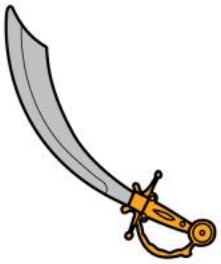


Figura 2.3: Pictograma que representa una espada.

levante. Por ejemplo, en la Figura 2.3 el objeto representado es una espada y el pictograma carece de elementos distractores como podría ser una mano que agarre la espada o un fondo que sirva de contexto.

Los pictogramas no son universales y no todo el mundo usa los mismos, existen diferentes colecciones. A continuación, presentamos algunas de las colecciones más conocidas y utilizadas.

2.2.1. Sistemas pictograficos

En esta sección se van a analizar algunos de los principales sistemas pictograficos.

2.2.1.1. Bliss

Bliss[4] fue desarrollado por Charles K. Bliss en 1949 y esta formado por más de 2.000 símbolos. Tiene la dificultad de que el usuario debe de conocer el sistema ya que los pictogramas de este sistema no cumplen la relación con el objeto representado tal y como se puede ver en la Figura 2.4, donde el pictograma animal, no guarda ninguna relación con aquello que representa

[4]https://www.uv.es/bellochc/logopedia/NRTLogo8.wiki?6

y dado solo el pictograma y sin conocer el sistema, no se puede conocer que objeto representa.

Figura 2.4: Picograma Bliss que representa un animal.

El sistema pictográfico Bliss (McDonald, 1985) fue creado con la idea de ser un lenguaje global. Está formado por un total de 26 letras en ingles y 28 para el castellano tal y como podemos ver en la Figura 2.5, además de un alfabeto de 26 símbolos básicos. La combinación de estas letras y estos símbolos básicos forman las palabras. En la Figura 2.6 podemos ver como el pictograma hogar es la combinación de los pictogramas para casa y sentimiento. La utilización de este sistema pictografico requiere de cierto nivel cognitivo (Hehner, 1985). Según el nivel de representación en Bliss los pictogramas se agrupan en tres grandes grupos:

1. Símbolos pictográficos: la forma del símbolo recuerda las palabras o conceptos que representa, esto lo podemos observar en el símbolo que representa una silla en la Figura 2.7.

2. Símbolos ideográficos: expresan una idea, no describen el objeto, por ejemplo en la Figura 2.8 podemos ver el símbolo de hogar, el cual tiene un corazón que significa sentimientos y que es no es algo tangible, con una casa.

3. Símbolos arbitrarios: expresan un concepto abstracto y el símbolo no guarda ningún relación con el objeto representado, como podemos observar en la Figura 2.9, donde aparece el pictograma para la conjunción "porque", que es un concepto abstracto.

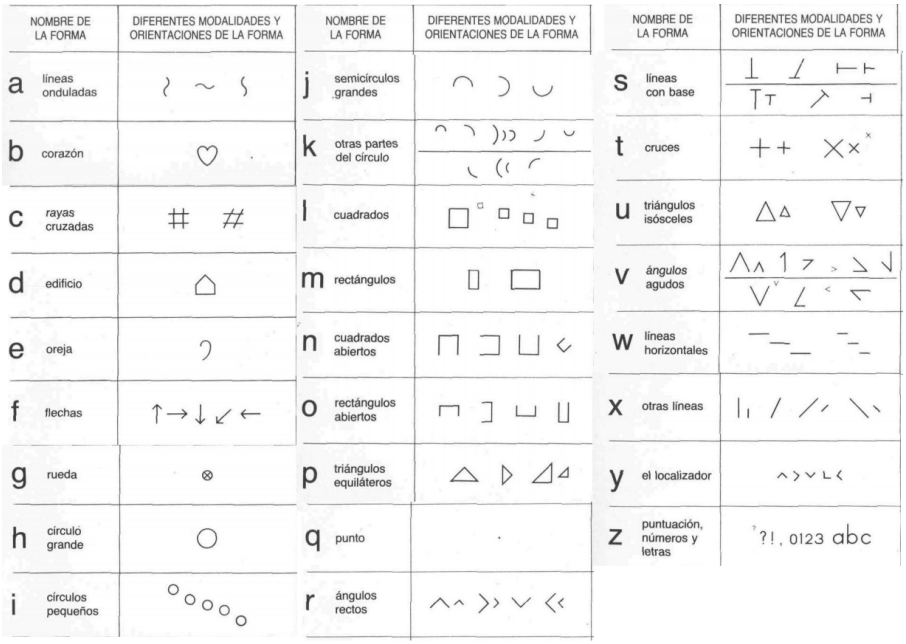


Figura 2.5: Todas las letras en castellano del sistema Bliss.

Figura 2.6: Combinación de pictogramas Bliss.

2.2.1.2. Minspeak

Minspeak[5] es un sistema pictográfico, que creo Bruce Baker en 1982 motivado por las carencias que observó en personas sin habla. Este sistema no

[5] https://minspeak.com/

Figura 2.7: Picograma Bliss que representa una silla.

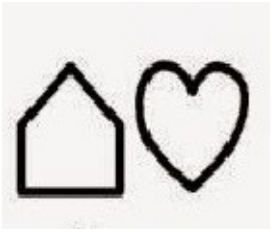


Figura 2.8: Creación de la palabra hogar en el sistema Bliss

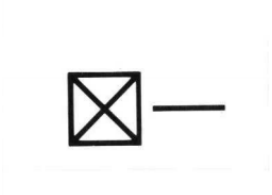


Figura 2.9: Pictograma Bliss que significa porque.

da un significado único a cada pictograma, si no que el significado se define entre el logopeda y el usuario[6]. De ahí que para un usuario, el pictograma que representa un arco iris puede significar felicidad pero para otro pueda significar lluvia. Además, existen un conjunto de reglas definidas por el sistema como la agregación de pictogramas para alcanzar diferentes significados.

Bruce Baker se inspiro en el lenguaje chino donde cualquier frase puede tener 12 caracteres o menos, y de esa manera definió su sistema que se centra en la combinación de pictogramas (Marín y Pérez, 2003). Para representar una palabra o frase se combinan símbolos, normalmente dos o tres, usando un conjunto de reglas y patrones definidos por el sistema Minspeak. Como se puede observar en la Figura 2.10, combinando el símbolo que representa una casa con el de una cama se obtiene el significado de habitación. Se considera un sistema muy accesible dado que el usuario debe de aprender un número limitado de pictogramas que suelen ir desde los 40 hasta los 80 pictogramas, variando este numero dependiendo de las capacidades del usuario. Las palabras ademas quedan clasificadas por grupos semánticos (verbos,

[6] http://ares.cnice.mec.es/informes/18/contenidos/94.htm

adjetivos...).

Minspeak permite desarrollar un lenguaje a través de la construcción de frases o secuencias que son fáciles de entender, pero tiene la desventaja de que el usuario necesita recordar la secuencia de símbolos necesarios para la generación de la respuesta correcta.

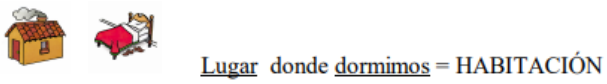


Figura 2.10: Sistema Minspeak.

2.2.1.3. SPC

El Sistema Pictografico de Comunicación (SPC)[7] fue desarrollado por Roxana Mayer-Jhonson en 1981. Busca a través de un diseño de pictrogramas simples, la similitud con aquello que representan. Los pictogramas en SPC están catalogados en seis grupos cada uno con un color para ayudar a la clasificación tal y como se puede ver en la Figura 2.11: personas en amarillo, verbos en verde, pictogramas descriptivos en azul, sustantivos en naranja, letras, números y miscelánea con fondo blanco y actos sociales en rosa. En total el sistema SPC lo forman un conjunto de 3.000 pictogramas. La comunicación mediante SPC carece de sintaxis lo que simplifica la construcción de frases.

2.2.1.4. ARASAAC

ARASAAC[8] surge en el año 2007 financiado por el Departamento de Educación Cultura y Deporte del Gobierno de Aragón y coordinado por la Dirección General de Innovación, Equidad y Participación de dicho departamento. Este proyecto nace con el objetivo de romper las barreras en la comunicación. ARASAAC cuenta con más de 33.000 pictogramas y con traducciones en más de quince idiomas entre ellos el castellano.

Actualmente, ARASAAC se ha convertido en un sistema de pictogramas reconocido internacionalmente, y sus recursos están bajo la licencia Creative Commons.

Los pictogramas están clasificados en cinco grandes grupos: pictogramas a color, en blanco y negro, fotografías y por ultimo fotografías y vídeos en Lengua de Signos Español ordenados dependiendo de su naturaleza semántica (Bertola López, 2018). Es importante destacar que también existen

[7] http://masquemayores.com/magazine/psicologia/tipos-de-sistemas-alternativos-y-aumentativos-de-la-comunicacion-sistemas-pictograficos-de-comunicacion/

[8] `http://www.arasaac.org`

Figura 2.11: Clasificación de los pictogramas SPC. Fuente: Universidad de Valencia Logopedia Audición y Lenguaje.

pictogramas para algunas conjunciones y determinantes tal y como podemos ver en la Figura 2.12 .

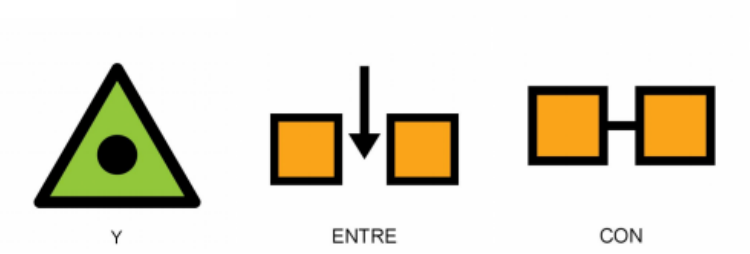


Figura 2.12: Ejemplos de conjunciones en ARASAAC.

A diferencia de otros sistemas pictograficos, ARASAAC hace distición de genero y numero tal y como se ve en la Figura 2.13, que muestra los distintos pictogramas para la palabra profesor, según el genero y número.

Además, ARASAAC permite añadir un marco al borde del pictograma, dicho marco sirve para clasificar la palabra diferenciando entre sujetos, verbos y objetos tal y como podemos ver en la Figura 2.14. Los sujetos no llevarían borde, las acciones o verbos tendrán un borde verde y en los objetos o complementos directos el borde sera amarillo.

En ARASAAC existen también pictogramas más complejos que expresan frases hechas tal y como podemos observar en el pictograma asociado a la acción *"sacar al perro"* en la Figura 2.15.

Figura 2.13: Ejemplo del pictograma profesor con diferente genero y numero

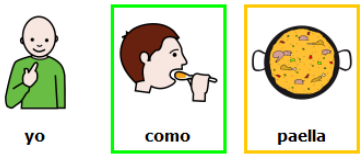


Figura 2.14: Ejemplo de bordes de color en pictogramas

Figura 2.15: Ejemplo de un pictograma complejo

En ARASAAC hay palabras que pueden tener varios pictogramas asociados, por ejemplo tal y como se puede observar en la Figura 2.16, la palabra *"teléfono"* tiene diferentes pictogramas asociados, y el uso de un pictograma u otro dependerá del nivel cognitivo del usuario, el contexto de la frase, etc. En ARASAAC los verbos carecen de conjugación como podemos observar en la Figura 2.17 donde se puede ver que el pictograma asociado a las diferentes conjugaciones del verbo ir es el mismo. Los tiempos verbales de una frase (presente, pasado o futuro) quedan determinados por los pictogramas de ayer, hoy o mañana, que podemos observar en la Figura 2.18.

Figura 2.16: Pictogramas asociados a la palabra “teléfono”

Figura 2.17: Ejemplo de verbos conjugados

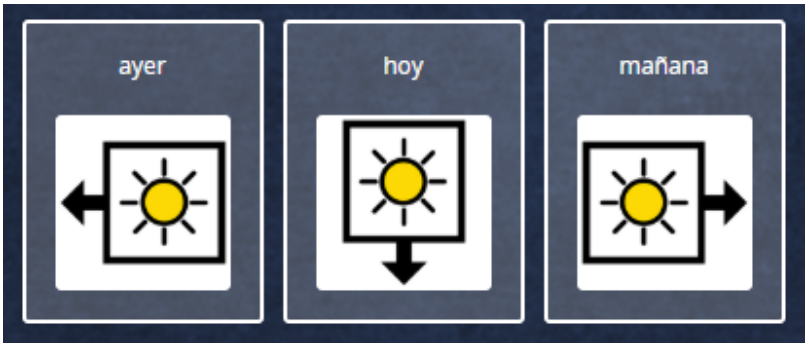


Figura 2.18: Pictogramas de hoy, mañana, y pasado

2.2.2. Comunicación a través de pictogramas

Comúnmente la comunicación con pictogramas se realiza a través de tableros, de manera que el usuario señala uno a uno los pictogramas en dicho tablero hasta completar la frase. En la Figura 2.19 podemos observar un tablero de ejemplo con pictogramas de ARASAAC para las actividades desarrolladas en un colegio (leer, hacer un examen, dibujar, ejercicios de caligrafía...). En este tablero además de los pictogramas asociados a cada actividad hay dos pictogramas uno que representa un pulgar hacía arriba y otro hacía abajo, que sirven para comunicar si o no respectivamente.

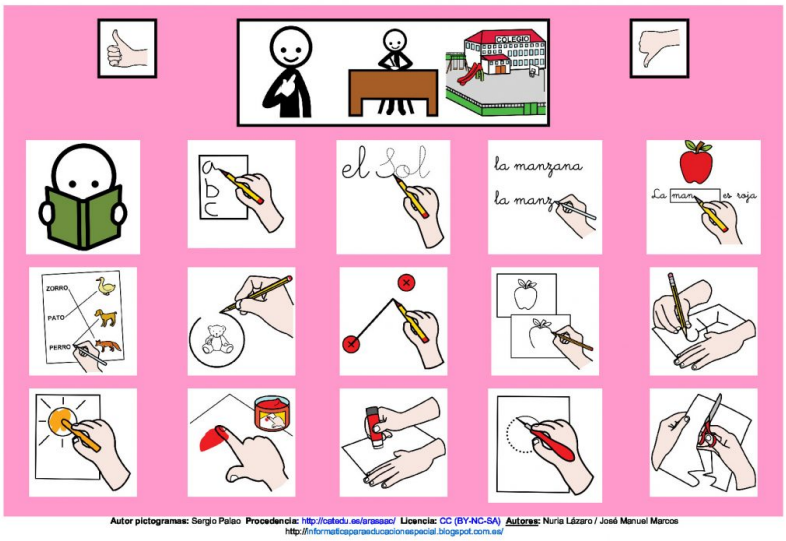


Figura 2.19: Tablero ARASAAC para actividades escolares.

Las frases con pictogramas suelen tener una complejidad reducida, limitándose casi siempre a sujeto, verbo y objeto, utilizando solo palabras significativas. Determinantes, conjunciones y preposiciones no suelen tener cabida dentro de la comunicación con pictogramas, aunque algunos sistemas pictograficos como ARASAAC, si que poseen pictogramas para representarlas.

Para fijar la temporalidad de la frase dentro de los sistemas pictograficos, existen conjunciones temporales que sirven para aportar dicha información. Por ejemplo, si en una frase aparece el pictograma "antes", el tiempo verbal de la frase será pasado. Además de conjugaciones, existen palabras que ayudan a determinar el tiempo verbal de la frase como pueden ser ayer, hoy o mañana.

2.3. Sistema de traducción texto-picto

Es muy importante destacar que no se ha encontrado ninguna herramienta que haga la traducción de picto a texto que se planea realizar en este trabajo. Por ello, en esta sección se presentan los sistemas de traducción texto a picto existentes, estos sistemas hacen la traducción inversa a lo que se pretende hacer en este TFG y nos ayudarán tomándolos como base para realizar nuestro trabajo.

2.3.1. Araword

Araword[9] es una aplicación gratuita que utiliza los pictogramas propios de Arasacc, y que permite la traducción simultanea de texto a pictograma. Esta herramienta tiene como objetivo principal facilitar la elaboración de materiales con pictogramas. Es similar a un editor de textos, se comienza con un documento en blanco y se van escribiendo frases en lenguaje natural que se van traduciendo a pictogramas. Cada vez que se introduce una palabra en el editor, la propia herramienta es la que se encarga de traducirla a pictograma automáticamente. Cabe destacar que esta aplicación tiene traducción simultanea, esto significa que a medida que se escribe la palabra el pictograma va cambiando tal y como se aprecia en la Figura 2.20

Figura 2.20: Ejemplo de transcripción simultanea en Araword.

Como podemos observar en los ejemplos, Araword inserta debajo de cada pictograma la palabra del texto a la que corresponde dicho pictograma. Además la aplicación no traduce las distintas conjugaciones empleadas en las frases, sino que para cualquier conjugación de un verbo emplea el pictograma asociado al infinitivo del verbo, tal y como puede verse en la Figura 2.21.

Es importante añadir que Araword no hace una traducción literal, es decir, la aplicación no traduce palabra por palabra sino que valora el sentido de la frase, y la posibilidad de traducir expresiones que engloban varias palabras con un solo pictograma, como podemos ver en la Figura 2.22, en la que la expresión "pasear al perro" ha sido traducida con un solo pictograma.

[9] http://www.arasaac.org/software.php

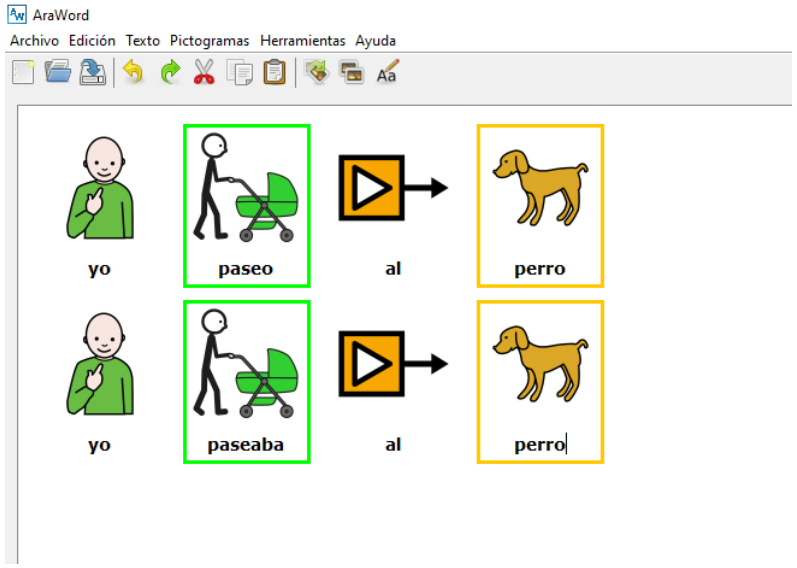


Figura 2.21: Ejemplo de traducción de la misma frase con diferentes tiempos verbales en Araword.

Figura 2.22: Ejemplo de traducción literal en Araword

2.3.2. PictoTraductor

PictoTraductor[10] es una aplicación web pensada y desarrollada para facilitar la comunicación mediante pictogramas. Principalmente está dirigida a padres y profesionales, para traducir textos a pictogramas y así poder co-

[10] https://www.pictotraductor.com/

municarse con sus hijos o pacientes que solo entienden pictogramas y no lenguaje natural. La interfaz de la aplicación se puede ver en la Figura 2.23. Consta de una barra de búsqueda en la que se introduce el texto que se quiere traducir a pictogramas. A medida que se va escribiendo una palabra se va mostrando el pictograma que representa el texto introducido en la parte inferior. Si las palabras tienen más de un pictograma asociado, la herramienta da al usuario la opción de elegir el pictograma mediante las flechas que aparecen encima del picto traducido. Para los artículos, PictoTraductor presenta por defecto el pictograma con la palabra escrita literalmente, dejando la opción al usuario de cambiar el pictograma a otro cuyo significado sea el mismo mediante las flechas dispuestas arriba y abajo de la imagen, tal y como vemos en la Figura 2.24.

Figura 2.23: Interfaz de PictoTraductor.

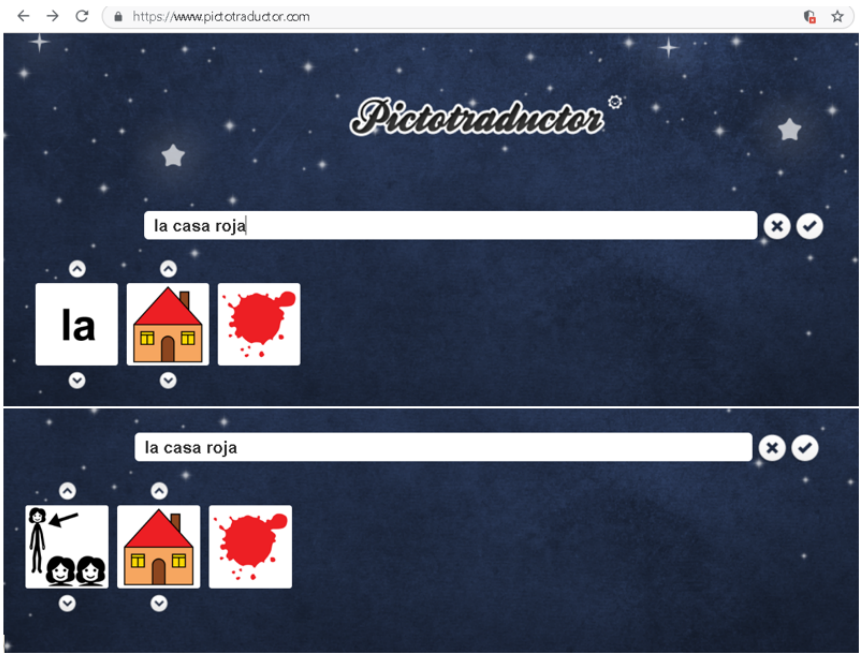


Figura 2.24: Ejemplo traduccíon de los artículos en PictoTraductor.

PictoTraductor no hace traducción literal de las frases ya que no realiza una traducción palabra por palabra, sino que busca el sentido de dicha frase y la posibilidad de traducir expresiones que engloban varias palabras con un solo pictograma, tal y como podemos observar en la Figura 2.25.

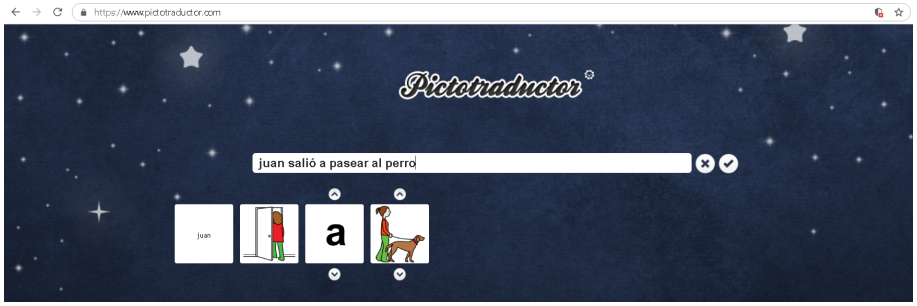


Figura 2.25: Ejemplo traducción literal en PictoTraductor.

Por último, la traducción de los diferentes tiempos verbales en las frases es siempre la misma, el infinitivo, tal y como se puede ver en la Figura 2.26.

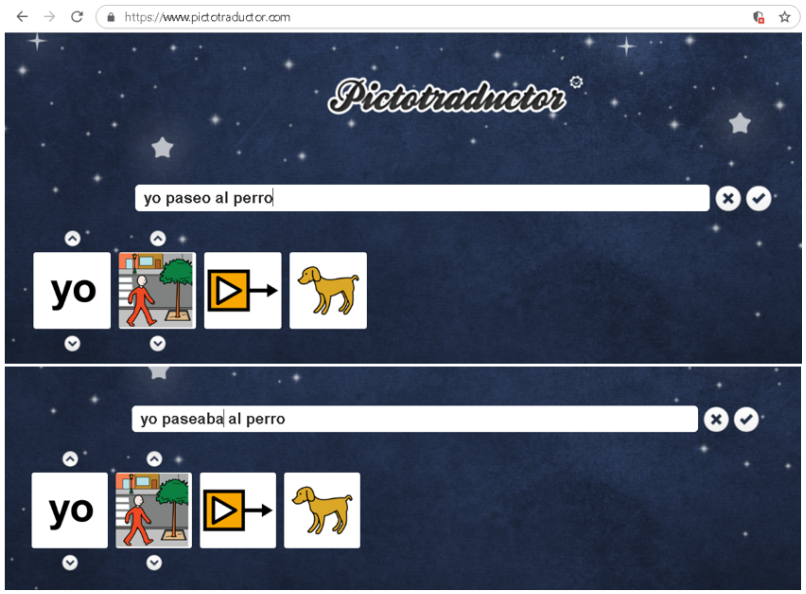


Figura 2.26: Ejemplo de traducción de la misma frase con diferentes tiempos verbales en PictoTraductor.

2.3.3. Pictar

Pictar (Martín Guerrero, 2018) es una aplicación web que permite traducir de texto a pictogramas. El usuario introduce el texto que desea traducir y al pulsar el botón buscar se generan los pictogramas adecuados para la frase introducida, tal y como se puede ver en la Figura 2.27. Esta aplicación no realiza una traducción literal de la frase, es decir, no hace una traducción palabra por palabra de la frase sino que analiza el sentido de la frase y busca pictogramas para expresiones, tal y como se puede ver en la Figura 2.28.

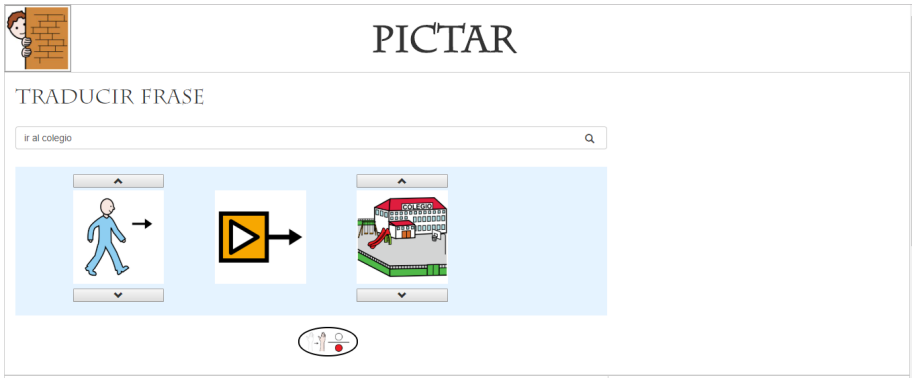


Figura 2.27: Texto traducido en Pictar.

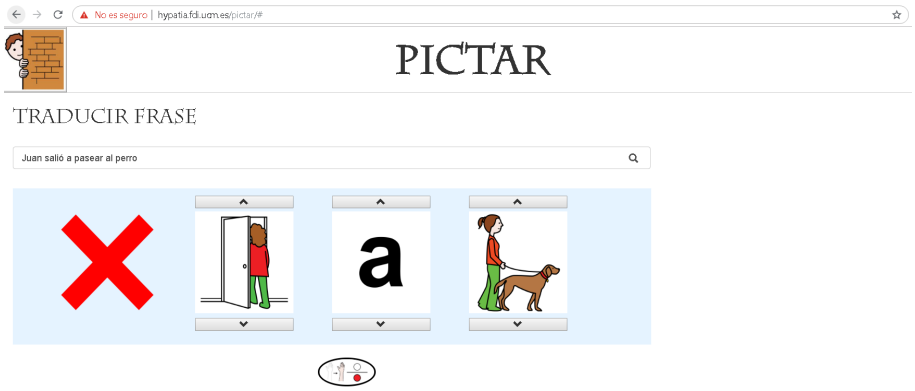


Figura 2.28: Ejemplos de traducción literal en Pictar.

Es importante destacar que Pictar no traduce de diferente manera las frases en las que se emplean distintos tiempos verbales, es decir en todas las traducciones el pictograma utilizado para el verbo será el infinitivo tanto para pasado, presente o futuro, podemos observarlo en el ejemplo que se muestra en la Figura 2.29.

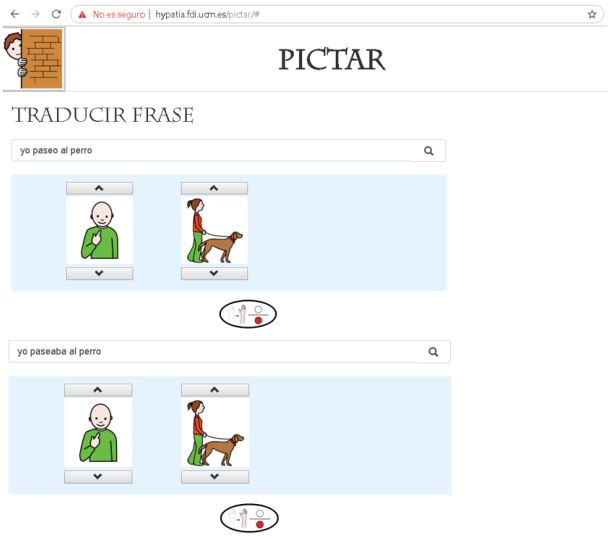


Figura 2.29: Ejemplo de traducción de la misma frase con diferentes tiempos verbales en Pictar.

Esta aplicación además ofrece en la parte derecha un buscador de pictogramas. El usuario introduce la palabra a buscar y en la parte inferior aparecen los pictogramas asociados a la palabra. El usuario puede mover los pictogramas encontrados a la rejilla de traducción. Pudiendo cambiar el color del pictograma, añadir texto debajo de él, etc, tal y como se puede ver en la Figura 2.30.

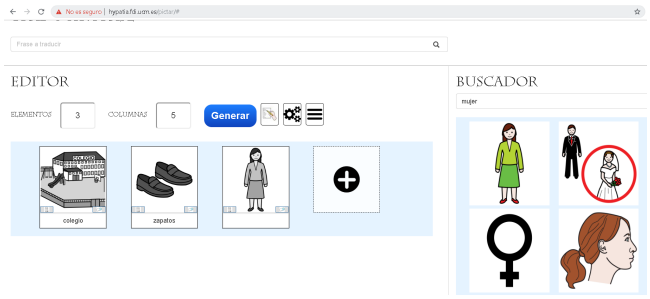


Figura 2.30: Ejemplo buscador y editor Pictar

2.4. Generación de Lenguaje Natural

La Generación de Lenguaje Natural (GLN) es un campo de la Inteligencia Artificial para crear programas informaticos que generan lenguaje natural ya sea hablado o escrito (Socorro Bernardos Galindo, 2007). Se busca que el programa generador se comunique igual que si de una persona se tratase (Dale et al., 2000; Vicente et al., 2015). La GLN se engloba dentro de la lingüística computacional aunque engloba muchas otras áreas de estudio fuera de la informática como la lingüística o la psicología.

Falsamente podemos pensar que lo más importante de la generación del lenguaje es la obtención de un texto gramaticalmente correcto. Pero, lo más importante es que el texto generado explique aquello que se desea transmitir. Por esa razón podemos concluir que el proceso de generación de lenguaje natural se podría resumir en la construcción de un mensaje que transmite una idea de manera clara.

Existe dos maneras de que un programa informático se comunique con un usuario. La primera forma es a través de mensajes estáticos definidos en el código, este acercamiento si bien es valido, genera un sistema cerrado y no flexible en el cual no hay lugar para la interpretación del lenguaje. La segunda vía, es la generación de lenguaje basada en conocimiento o *deep generation* donde es el propio sistema el encargado de generar el lenguaje basándose en el conocimiento lingüístico del que dispone (García Ibáñez et al., 2004).

Durante la generación del lenguaje se puede abstraer el proceso y encapsularlo en diferentes fases tal y como se puede ver en la Figura 2.31, dichas fases serán descritas brevemente a continuación (Vicente et al., 2015):

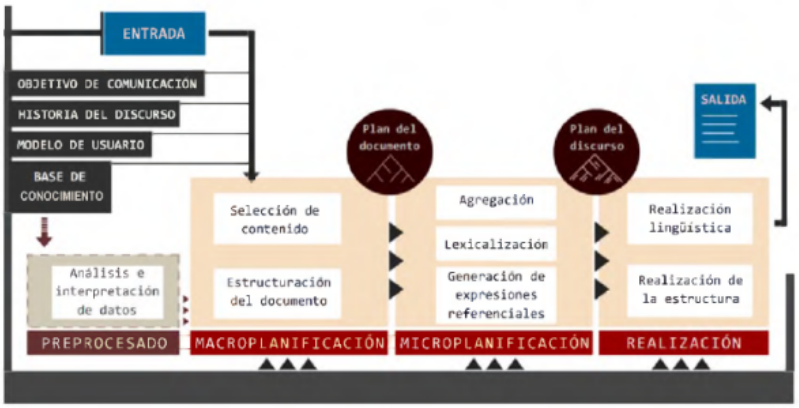


Figura 2.31: Fases de generación del lenguaje. Fuente: `http://www.scielo.org.mx`.

1. Preprocesado. Esta etapa es opcional y puede no aparecer en un sistema de GLN, dado que solamente es necesaria cuando existen datos que han de ser analizados e interpretados. El objetivo del preprocesado es la generación de estructuras de datos que permitan que el texto de entrada sea más fácil de tratar en las siguientes fases (Han, 2005). En la Figura 2.32, podemos ver una entrada de datos meteorológicos que necesitan un preprocesamiento, antes de poder añadirse al texto.

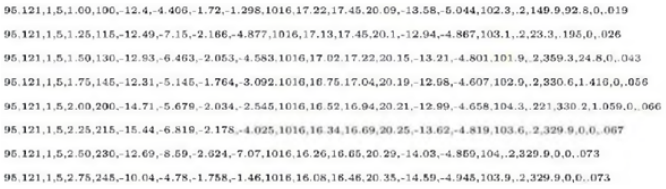


Figura 2.32: Ejemplo de una entrada de texto con datos meteorológicos, para la fase de preprocesado.

2. Macro planificación. Etapa donde se debe seleccionar el contenido que se desea convertir a texto. Para que el texto generado sea coherente es preciso estructurar el contenido del mismo en el orden correcto. La salida de esta fase es el denominado plan del documento, una estructura de datos que suele ser un árbol, donde en cada nodo se encapsula la información más importante que debe formar parte de un párrafo o de una frase, además de información de como se relaciona con el resto de nodos. En la Figura 2.33 podemos ver un ejemplo de la salida de esta etapa, en la que se puede ver una estructura del párrafo obtenido tras terminar la macro planificación, en el árbol resultante podemos observar la secuencia temporal que tendrá el párrafo.

Figura 2.33: Árbol resultado de la fase de macro planificación para la recogida de datos meteorológicos.

3. Micro planificación. Partiendo de la salida de la macro planificación, se determina que partes formaran el texto resultado, agregando las estructuras que formaran parte de este, el léxico que se utilizara para expresar los conceptos y los hechos recogidos en la primera etapa y la generación de expresiones que aparecerán en el texto. Siguiendo el ejemplo visto en la macro planificación se pueden generar pequeñas frases referenciales como pueden ser "Viento de las 6:00" o "a las 15:00 el viento cambio", donde el referente común es "el viento".

4. Realización. El objetivo de esta fase es la creación de las oraciones finales y de la estructura final que tendrá el texto. La generación o realización del texto se puede dividir en dos partes:

   - Realización Lingüística: en esta parte se determina la representación en palabras del texto de salida, es decir, se forman las oraciones. Durante esta fase se hace uso de un diccionario de palabras para la resolución de las diferentes formas de las palabras como puede ser el genero o el número. Cada palabra del diccionario suele tener la siguiente estructura:
     - Forma base de la palabra.
     - Categoría Léxica (nombre, adjetivo, verbo, determinante...).
     - Genero (masculino, femenino o indeterminado).
     - Número (singular o plural).

     Dependiendo del tipo de palabra pueden aparecer otras características. En caso de un verbo el diccionario también tiene las diferentes conjugaciones para los tiempos verbales (presente, pasado, futuro, imperativo...). En caso de palabras como nombres, determinantes o adjetivos, se añaden además las siguientes entradas al diccionario:
     - Palabra en singular.
     - Palabra en plural.
     - Palabra con genero femenino.
     - Palabra con genero femenino y en plural.
   - Realización de la Estructura: en esta parte se define el formato correcto del texto generado para su visualización en un medio determinado, como puede ser HTML, o un documento Latex, tal y como podemos ver en la Figura 2.34.

Existen diferentes tipos de arquitectura de sistemas de GLN (García Ibáñez et al., 2004):

1. Integrada o monolítica: el sistema es un solo bloque sin división clara entre módulos.

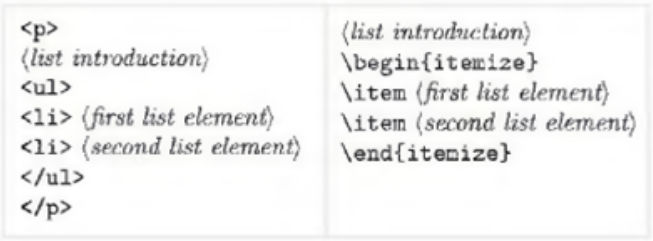


Figura 2.34: Salida de la fase de la realización donde podemos ver la estructura final en HTML a la izquierda y Latex a la derecha.

2. Secuencial: los módulos se encadenan en un flujo unidireccional.

3. Interactiva: se permite la revisión de las decisiones tomadas. La información puede circular tanto hacía delante como hacía atrás.

4. Pizarra: la información no va de modulo a modulo si no que se guarda en una zona común, de manera que cada modulo pueda modificarla.

5. Basada en revisión: la información fluye de forma cíclica entre los módulos hasta alcanzar un estado óptimo.

2.5. Servicios web

El WC3 (World Wide Web Consortium) define un Servicio Web como[11]:

Un sistema software diseñado para soportar la interacción máquina-a-máquina, a través de una red, de forma interoperable. Cuenta con una interfaz descrita en un formato procesable por un equipo informático (específicamente en WSDL), a través de la que es posible interactuar con el mismo mediante el intercambio de mensajes SOAP, típicamente transmitidos usando serialización XML sobre HTTP conjuntamente con otros estándares web.

Un Servicio Web es un componente accesible mediante peticiones Web estándar tales como GET o POST. Normalmente se denomina servicio web a una colección de procedimientos accesibles desde Internet. La gran ventaja de los servicios web es que al acceder desde el navegador a un servicio web este nos devuelve el contenido con un formato estándar como puede ser XML.

Estos servicios web se empezaron a desarrollar para solucionar el problema de la interoperabilidad entre las distintas aplicaciones. En la década de los 90, el objetivo principal era poder integrar diferentes aplicaciones dentro

[11] https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211

del mismo servicio. Estas aplicaciones estaban desarrolladas en muchos lenguajes de programación como: C++, Java, etc. Y podían ejecutarse tanto en Unix, en un PC, o un computador mainframe. Por todas estas diferencias, era muy difícil y costoso comunicar las aplicaciones entre sí. Gracias al desarrollo de XML, se consiguió poder compartir datos entre ellas a través de la red, o de Internet.

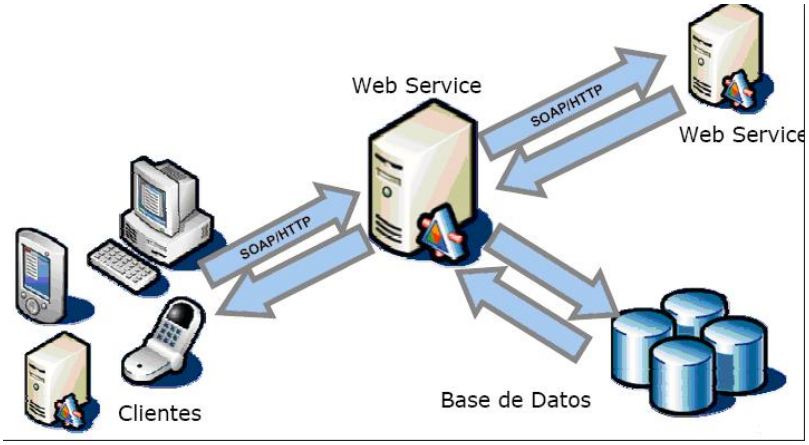


Figura 2.35: Estructura servicios Web. Fuente: `https://sites.google.com/site/preyectodetics/home/servicios-web`

A nivel técnico, los servicios web más usados suelen ser de dos tipos.

- SOAP. Utilizan el lenguaje XML para poder comunicarse utilizando el protocolo SOAP (Simple Object Access Protocol), este lenguaje es usado tanto para definir la arquitectura como el formato de los mensajes. Estos tipos de servicios deben contener un lenguaje legible para la máquina con la descripción de operaciones ofrecidas por el servicio escrito en WSDL (Web Services Description Language), el cuál está basado también en XML con el fin de definir interfaces sintácticamente.

- Restfull (Representational State Transfer Web Services). Los servicios web Restfull utilizan estándares como: SML, URI, MIME, y sobre todo HTML. Además posee una infraestructura que podríamos llamar ligera, ya que utilizan las herramientas mínimas para formar los servicios. El desarrollo de este tipo de servicios es barato y su adaptación poco costosa. Son más adecuados para integraciones básicas ad-hoc. Estos servicios suelen mezclarse mejor con HTTP que los basados en SOAP, ya que no necesitan mensajes XML ni definiciones WSDL.

En una arquitectura de servicios web se pueden distinguir tres partes

fundamentales para el establecimiento de la comunicación[12]:

- Proveedor de servicios web. Es el encargado de enviar al publicador del servicio un fichero WSDL con la información básica de dicho servicio.
- Publicador del servicio. Una vez la información llega al publicador este descubre quien es el proveedor (protocolo WSDL), y se pone en contacto con él (protocolo SOAP y REST).
- Proveedor del servicio. Acepta la petición del servicio y envía los datos estructurados en formato XML usando SOAP, si fuese un servicio REST los datos enviados serían de tipo JSON.

2.5.1. Ventajas de los servicios web

Las tres principales ventajas de usar servicios web son[13]:

- Ayudan a mejorar la conexión y operabilidad entre dos aplicaciones software sin importar sus propiedades o sino comparten plataforma de instalación.
- Favorecen los protocolos y estándares basados en texto, ya que estos facilitan acceder y comprender su contenido.
- Permiten y facilitan que software y servicios de diferentes compañías se puedan combinar formando un servicio funcional.

2.5.2. Desventajas de los servicios web

Las tres principales desventajas de los servicios web son[14]:

- Las transacciones que llevan consigo estos servicios están mucho menos desarrolladas que otros estándares abiertos.
- Su rendimiento es bajo comparado con otros modelos de computación distribuida, debido sobre todo a adoptar un formato basado en texto.
- Al estar muy ligados con HTTP, las medidas de seguridad que tratan de bloquear la comunicación entre programas se pueden burlar fácilmente.

[12] http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html
[13] https://sites.google.com/site/preyectodetics/home/servicios-web
[14] http://fabioalfarocc.blogspot.com/

Capítulo 3

Herramientas

A lo largo de este capítulo analizaremos las diferentes herramientas utilizadas para el desarrollo de este TFG. Para implementar todas las funcionalidades de la aplicación vamos a utilizar servicios web para traducir un texto escrito con pictogramas a lenguaje natural, lo primero es conseguir las palabras asociadas a cada pictograma del mensaje de entrada para ello haremos uso de la API de Arasaac, que explicaremos en la Sección 3.1. Para generar frases gramaticalmente correctas a partir de las palabras asociadas a los pictogramas vamos a necesitar un generador de lenguaje natural como es SimpleNLG (Sección 3.2) y un analizador sintáctico como Spacy (Sección 3.3). También, necesitamos un espacio donde conectar los diferentes servicios web, para ello utilizamos Django, que será explicado en la Sección 3.4. Por último, para agilizar tanto el tiempo de respuesta de los servidores como el proceso de comunicación de los diferentes servicios web se ha elegido la herramienta Angular (Sección 3.5) para implementar el Front-End de nuestra aplicación.

3.1. API-ARASAAC

En nuestra aplicación los pictogramas son fundamentales y necesitábamos una herramienta que nos permitiese:

- Obtener los pictogramas asociados a una determinada palabra.
- Obtener la palabra o palabras asociadas a un pictograma.

Para ello hemos empleado la API de ARASAAC[1], ya que de todos los sistemas pictográficos vistos en la Sección 2.2, los pictogramas de Arasaac son los más utilizados en los países hispanohablantes. Además, Arasaac ofrece multitud de herramientas gratuitas.

[1] https://beta.arasaac.org/developers/api

De todos los métodos ofrecidos por la API de ARASAAC hemos usado los siguientes:

- SearchText[2]: Dada una palabra devuelve un JSON con el tipo de palabra, su significado, sus plurales y el identificador del pictograma que tiene asociado. Tal y como se puede ver en la Figura 3.1 con la palabra "perro". Si una misma palabra tiene distintos significados o distintos pictogramas asociados, en el JSON devuelto aparecerán las diferentes opciones, tanto si tiene varios pictogramas asociados como si tiene distintos significados, como podemos ver en la Figura 3.2

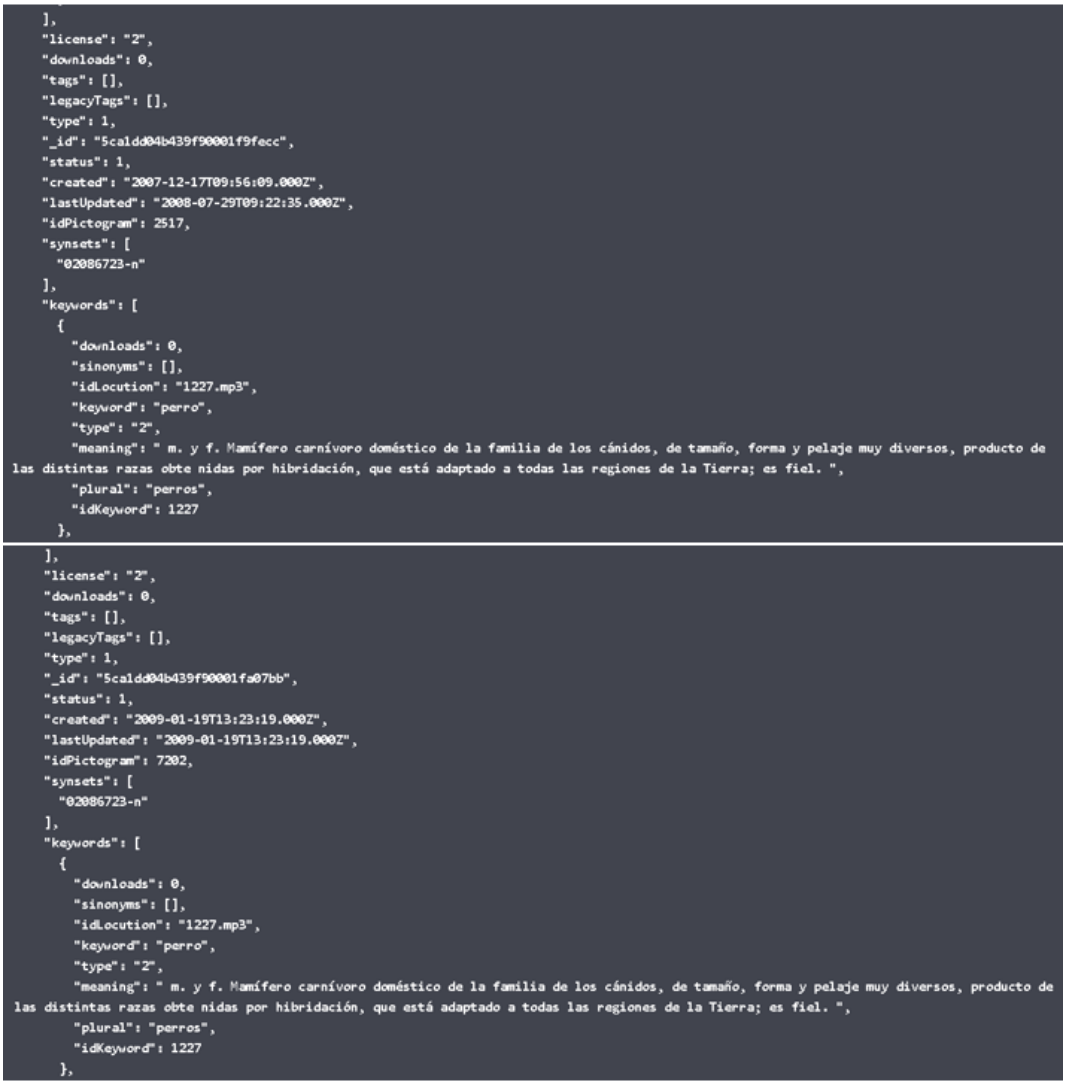

```
  ],
  "license": "2",
  "downloads": 0,
  "tags": [],
  "legacyTags": [],
  "type": 1,
  "_id": "5ca1dd04b439f9001f9facc",
  "status": 1,
  "created": "2007-12-17T09:56:09.000Z",
  "lastUpdated": "2008-07-29T09:22:35.000Z",
  "idPictogram": 2517,
  "synsets": [
    "02086723-n"
  ],
  "keywords": [
    {
      "downloads": 0,
      "sinonyms": [],
      "idLocution": "1227.mp3",
      "keyword": "perro",
      "type": "2",
      "meaning": " m. y f. Mamífero carnívoro doméstico de la familia de los cánidos, de tamaño, forma y pelaje muy diversos, producto de
las distintas razas obte nidas por hibridación, que está adaptado a todas las regiones de la Tierra; es fiel. ",
      "plural": "perros",
      "idKeyword": 1227
    },
```


Figura 3.1: JSON devuelto por el método SearchText para la palabra perro. Fuente: https://beta.arasaac.org/developers/api.

- idPictogram[3]: Dado un identificador de un pictograma devuelve un JSON con toda la información asociada a la palabra asociada al pictograma con dicho id. Por ejemplo con el id 2549 se obtiene un JSON con la información de la palabra *reloj*, tal y como se muestra en la Figura 3.3

La API de ARASAAC es gratuita con fines no comerciales, siempre que en la aplicación se mencione explícitamente a ARASAAC y se cumpla la licencia CC (Creative Commons) RY-NC-SA[4].

[2] `https://beta.arasaac.org/developers/api/pictograms/search/searchText`
[3] `https://beta.arasaac.org/developers/api/pictograms/idPictogram`
[4] https://creativecommons.org/licenses/by-nc-sa/3.0/es/



```
    ],
    "license": "2",
    "downloads": 0,
    "tags": [],
    "legacyTags": [],
    "type": 1,
    "_id": "5calda04b439f90001f9fecc",
    "status": 1,
    "created": "2007-12-17T09:56:09.000Z",
    "lastUpdated": "2008-07-29T09:22:35.000Z",
    "idPictogram": 2517,
    "synsets": [
      "02086723-n"
    ],
    "keywords": [
      {
        "downloads": 0,
        "sinonyms": [],
        "idLocution": "1227.mp3",
        "keyword": "perro",
        "type": "2",
        "meaning": " m. y f. Mamífero carnívoro doméstico de la familia de los cánidos, de tamaño, forma y pelaje muy diversos, producto de
las distintas razas obte nidas por hibridación, que está adaptado a todas las regiones de la Tierra; es fiel. ",
        "plural": "perros",
        "idKeyword": 1227
      },
    ],
    "license": "2",
    "downloads": 0,
    "tags": [],
    "legacyTags": [],
    "type": 1,
    "_id": "5calda04b439f90001fa07bb",
    "status": 1,
    "created": "2009-01-19T13:23:19.000Z",
    "lastUpdated": "2009-01-19T13:23:19.000Z",
    "idPictogram": 7202,
    "synsets": [
      "02086723-n"
    ],
    "keywords": [
      {
        "downloads": 0,
        "sinonyms": [],
        "idLocution": "1227.mp3",
        "keyword": "perro",
        "type": "2",
        "meaning": " m. y f. Mamífero carnívoro doméstico de la familia de los cánidos, de tamaño, forma y pelaje muy diversos, producto de
las distintas razas obte nidas por hibridación, que está adaptado a todas las regiones de la Tierra; es fiel. ",
        "plural": "perros",
        "idKeyword": 1227
      },
```


Figura 3.2: JSON devuelto por el método SearchText para la palabra perro con dos pictogramas asociados. Fuente: `https://beta.arasaac.org/developers/api`

3.2. SimpleNLG

Para la aplicación se necesitaba un generador de frases gramaticalmente correctas en castellano. La única herramienta que cumplía con estos requisitos era SimpleNLG, una biblioteca hecha en Java que permite la generación de lenguaje natural (Jozef Trzpis, 2015). Simple-NLG permite crear frases simples y dentro de estas se pueden crear tanto oraciones nominales, como por ejemplo “la casa azul” u oraciones verbales, como “El perro era negro”.

```
],
"license": "2",
"downloads": 0,
"tags": [],
"legacyTags": [],
"type": 1,
"_id": "5ca1dd04b439f90001f9feea",
"status": 1,
"created": "2007-12-17T13:09:14.000Z",
"lastUpdated": "2008-07-31T10:28:39.000Z",
"idPictogram": 2549,
"synsets": [
  "04615227-n",
  "04563183-n",
  "03050242-n"
],
"keywords": [
  {
    "downloads": 0,
    "sinonyms": [],
    "idLocution": "931.mp3",
    "keyword": "reloj",
    "type": "2",
    "meaning": " m. Máquina que sirve para medir el tiempo o dividir el día en horas, minutos y segundos: reloj de pulsera, de cuco. ",
    "plural": "relojes",
    "idKeyword": 931
  }
]
```


Figura 3.3: JSON devuelto por el método idPictogram para el identificador 2549. Fuente: https://beta.arasaac.org/developers/api.

Dependiendo del tipo de frase, la oración se forma de una u otra forma:

- **Oraciones nominales**. Cuando la frase no tiene ningún verbo, SimpleNLG comienza generando el sustantivo de la frase y añadiendo a dicho sustantivo el determinante que concuerde con él en género y número. Para ello se le debe proporcionar a SimpleNLG el sustantivo de la oración como dato de entrada y la herramienta añadirá automáticamente el determinante que concuerde en género y número con él. Una vez creado el sustantivo con su determinante correspondiente, la herramienta permite añadirle complementos, como pueden ser: adjetivos, adverbios, etc. Por ejemplo, si queremos generar la frase "La casa azul" debemos proporcionar a SimpleNLG el sustantivo ("casa"), el género (femenino), el número (singular) y el complemento del sustantivo ("azul").

- **Oraciones verbales**. Estos tipos de oraciones tienen tres partes::

  - Sujeto: Se genera de la misma forma que en las frases nominales.
  - Verbo: Para generarlo debemos proporcionar a SimpleNLG el verbo en infinitivo y el tiempo verbal en el que hay que conjugarlo (pasado, presente o futuro). El género y el número para la conjugación del verbo lo obtendrá automáticamente SimpleNLG a partir de la información del sujeto.

- Complementos. Para generar el predicado debemos proporcionar a SimpleNLG las palabras que lo componen junto con su categoría léxica (sustantivo, adjetivo, adverbio...). Con toda esta información SimpleNLG genera un predicado gramaticalmente correcto.

Por ejemplo, para generar la frase "El perro era negro", deberíamos proporcionar a SimpleNLG el sujeto ("perro"), el verbo ("ser") con el tiempo verbal (pasado) y el complemento ("negro").

Por último, es importante destacar que SimpleNLG nos permite introducir negaciones en las oraciones junto con otras muchas funcionalidades que quedan fuera del ámbito de nuestro trabajo.

3.3. Spacy

Una vez que transformamos los pictogramas en palabras, se necesita conocer la categoría lexica de cada una de esas palabras (sustantivo, verbo, adjetivo...) así como su género y número para poder generar una frase gramatical correcta, ya que como hemos visto en la Sección 3.2, SimpleNLG necesita esta información. Para esta función de análisis usamos la librería Spacy. Esta elección se basó en que es la herramienta que posee una mayor precisión en la traducción, tal y como se puede observar en la Figura 3.4.

SYSTEM	YEAR	LANGUAGE	ACCURACY	SPEED (WPS)
spaCy v2.x	2017	Python / Cython	**92.6**	*n/a* ⓘ
spaCy v1.x	2015	Python / Cython	91.8	13,963
ClearNLP	2015	Java	91.7	10,271
CoreNLP	2015	Java	89.6	8,602
MATE	2015	Java	92.5	550
Turbo	2015	C++	92.4	349

Figura 3.4: Benchmark de diferentes procesadores del lenguaje. Fuente: https://spacy.io/.

Spacy es una biblioteca escrita en Python diseñada para el procesamiento avanzado de lenguaje natural[5]. La clase Tokenizer de Spacy es la encargada de dividir el mensaje recibido en palabras y de informarnos de la categoría (sustantivo, adjetivo, pronombre, verbo, determinante, adverbio, preposición, conjunción o complemento) de cada palabra, junto con las siguientes características según la categoría:

[5]https://medium.com/datos-y-ciencia/comenzando-con-spacy-para-procesamiento-de-lenguaje-natural-e8cf24a18a5a

- **Sustantivos** y **Adjetivos**. Género, número y sinónimos.
- **Pronombres**. Tipo de pronombre (personal, demostrativo, indefinido, posesivo)y número de la persona (si es un pronombre personal o posesivo devuelve el número de la persona a la que hace referencia).
- **Verbos**. Forma verbal (infinitivo, gerundio o participio)y tiempo verbal (pasado, presente o futuro).
- **Determinantes**. Tipo de determinante (artículo, posesivo, numeral), género y número.
- **Adverbios**. Tipo de adverbio (de lugar, de tiempo, de cantidad, de modo).

En la Tabla 3.1, podemos observar un ejemplo del análisis que realiza Spacy para la frase "Yo veo un perro rápido".

PALABRA	CAT. LÉXICA	ATRIBUTOS
Yo	Preposición	TipoPron: Personal, Person: Primera, Núm: Singular
veo	Verbo	FormV: Infinit., Modo: Indicativo, TiempoV: Presente, Person: Primera, Núm: Singular
un	Determinante	TipoDet: Artículo, Gen: Masculino, Núm: Singular
perro	Sustantivo	Gen: Masculino, Núm: Singular
rápido	Adjetivo	Gen: Masculino, Núm: Singular

Tabla 3.1: Ejemplo del análisis hecho por Spacy para la frase "Yo veo un perro rápido".

3.4. Django

Cada funcionalidad que se utiliza en este TFG ha sido implementada como un servicio web, por ello se decidió utilizar un framework que ayudase a los programadores a crear e implementar de una manera más fácil y eficiente dichos servicios web. Además estos servicios están escritos en Python ya que es el lenguaje de programación más usado para el por lo que es aconsejable que dicho framework estuviera también desarrollado en Python.

Con estas restricciones en mente se eligió la herramienta Django, la cuál nos permite desarrollar aplicaciones web[6] de código abierto y gratuitas escritas en Python.

[6] `https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/`

Django permite la creación de aplicaciones web. Este framework permite el desarrollo de aplicaciones web, usando el patrón Modelo-Vista-Template[7], el cual permite el uso de plantillas para la generación de vistas. Este patrón presenta el problema de que reduce la flexibilidad de acciones del lado del cliente, ya que por ejemplo cualquier modificación de la vista requiere de lanzar una petición al servidor y genere una vista nueva, en vez de modificar la actual. Además Django se puede utilizar para la creación de servicios web a través de objectos JSON[8]. Estas ventajas sumadas, a la comunidad de desarrolladores y la gran versatilidad de configuración que otorga Django (por ejemplo poder utilizar diferentes métodos de seguridad, manejar las rutas de los diferentes servicios web de una manera muy simple, manejar los diferentes archivos estáticos de la aplicación como puedan ser los scripts o las imágenes) han sido factores fundamentales para escoger Django en la implementación de nuestra aplicación.

Django se puede instalar utilizando el gestor de paquetes de python pip[9], una vez instalado permite mediante el uso de la linea de comandos[10], crear una aplicación Django[11] y servicios web utilizando distintos comandos.

3.5. Angular

Angular se ha usado para gestionar toda la parte Front-End de nuestra aplicación haciendo cada parte de la página web independiente. Angular es un framework de desarrollo creado y mantenido por Google, cuya finalidad consiste en el desarrollo de aplicaciones web SPA (Single-Page Applications), es decir, interacción web con una sola página, lo que provoca que la aplicación sea más rápida, dinámica y fluida.

Para realizar la parte Front-End de nuestra aplicación se estudiaron diferentes herramientas. En un primer momento se pensó en extender la herramienta Django para la parte del cliente, pero dicha herramienta está pensada principalmente para la gestión de la parte Back-End y por tanto se descartó esta idea. La segunda alternativa fue introducir JQuery. JQuery es una librería orientada a acceder y modificar los elementos de la página, pero carece de la facilidad que otorga Angular al utilizar componentes, aplicar patrones de arquitectura y sobretodo, carece de la rapidez y la facilidad de Angular para el tratamiento de llamadas asíncronas. Por estos motivos decidimos usar Angular, ya que era la alternativa que más funcionalidad, rapidez y escalibilidad, aportaba de las herramientas analizadas.

[7] https://docs.hektorprofe.net/django/web-personal/patron-mvt-modelo-vista-template/
[8] https://docs.djangoproject.com/en/2.2/ref/request-response/
[9] https://docs.python.org/3/installing/index.html
[10] https://www.python.org/shell/
[11] https://docs.djangoproject.com/en/2.2/intro/tutorial01/

Además, Angular nos permitía crear nuestra aplicación por componentes[12], ofreciendo así una división del código por funcionalidades también en la parte del Front-End. Los componentes tienen la ventaja de ser completamente independientes unos de otros y ser reutilizables, esta independencia que poseen los componentes otorgan una gran flexibilidad a la hora de construir aplicaciones, aumentan la escalabilidad y facilitan el mantenimiento.

Otro punto importante a favor del uso de Angular es que proporciona una actualización automática de la información, es decir, los datos mostrados en la vista están actualizados gracias al uso del modelo que proporciona Angular[13].

El problema que conlleva el uso de servicios asíncronos como las llamadas a servicios externos por HTTP es que no siguen un orden definido en el tiempo de respuesta de cada llamada. Angular resuelve este problema implantando el patrón 'Observer"[14]. La librería que usa Angular para utilizar este patrón es la RxJS (Reactive Extensions for JavaScript)[15], una librería centrada en el paradigma de la programación reactiva, la cual otorga diferentes ayudas para la resolución de llamadas asíncronas.

[12] `https://angular.io/guide/architecture-components`

[13] `https://angular.io/guide/displaying-data`

[14] `https://desarrolloweb.com/articulos/introduccion-teorica-observables-angular.html`

[15] https://angular.io/guide/rx-library

Capítulo 4

Pict2Text

A lo largo de este capitulo expondremos la aplicación Pict2Text que hemos desarrollado para este proyecto. Pict2Text (ver Figura 4.1) es una aplicación que permite la traducción de textos escritos con pictogramas a lenguaje natural. Pict2Text posee un buscador de pictogramas que sirve para obtener los pictogramas con los que construir el mensaje que se desea traducir. El buscador permite buscar los pictogramas de ARAASAC asociados a una determinada palabra. Cabe destacar que lo ideal sería poder cargar el mensaje en pictogramas a partir de un archivo o haciendo una foto al mensaje, en vez de construirlo mediante el buscador, pero dado que el foco de este TFG es la traducción se decidió implementar un buscador y dejar como trabajo futuro otras formas de introducir el texto en pictogramas para la traducción.

En la Sección 4.1 mostraremos la arquitectura de nuestra aplicación. En la Sección 4.2 detallaremos los diferentes servicios web implementados para dotar de funcionalidad a la aplicación en la parte Back-End. Por ultimo, explicaremos como hemos implementado la parte Front-End de la aplicación con el framework Angular, describiendo los diferentes componentes que la forman y la relación que guardan con la parte Back-End. En la Figura 4.2, podemos ver un diagrama en el que se muestran las partes Back-End y Fron-End de nuestra aplicación y su relación.

4.1. Arquitectura

El trabajo de este TFG está englobado dentro del proyecto nacional IDiLyCo[1]. IDiLyCo es un proyecto que busca facilitar la inclusión digital, de aquellas personas que por diversidad funcional, tienen problemas con el lenguaje natural. Uno de los objetivos del proyecto es desarrollar pequeñas

[1] http://nil.fdi.ucm.es/index.php?q=projects/idilyco

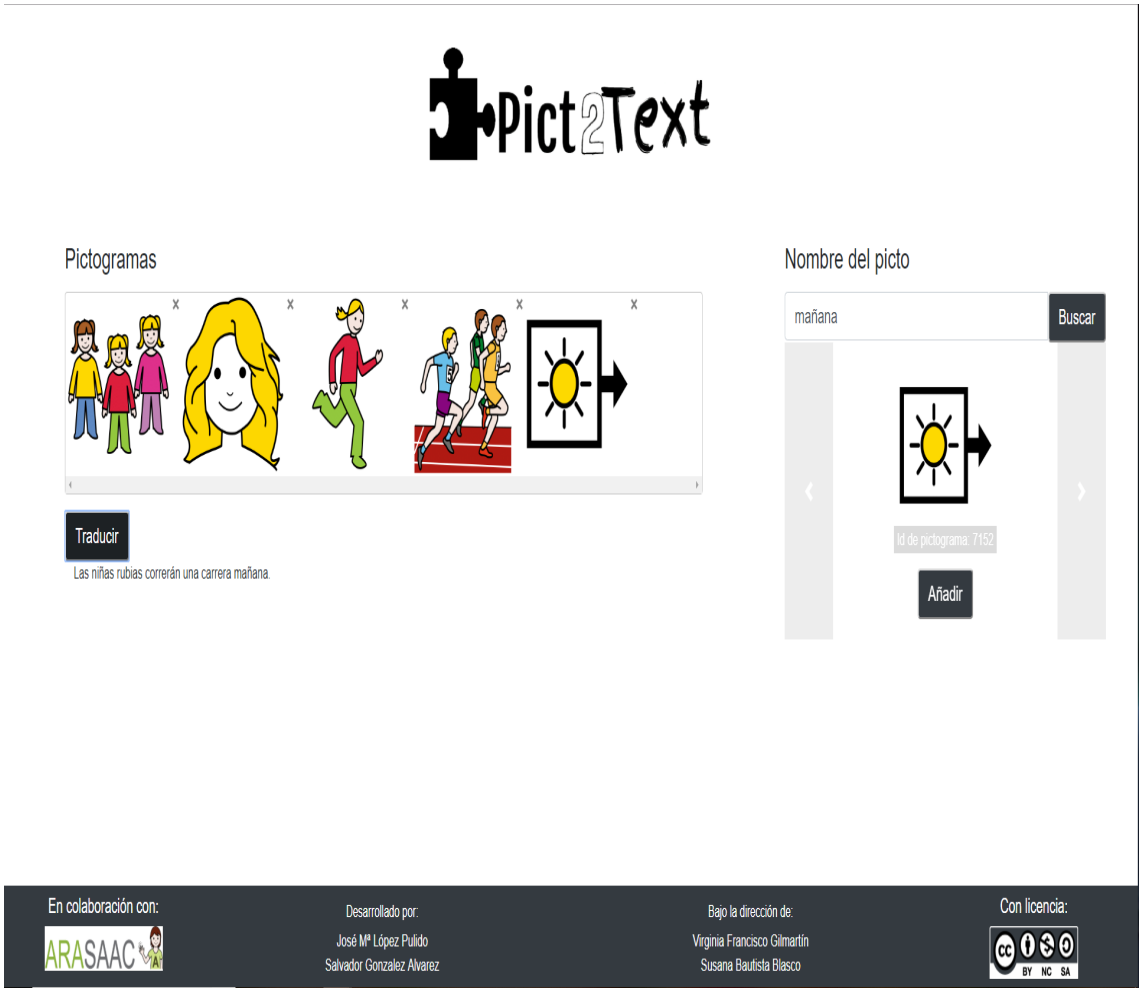


Figura 4.1: Pict2Text

piezas de funcionalidad reutilizables que se combinan en diferentes aplicaciones, es por ello que debíamos diseñar una arquitectura con muy bajo acoplamiento. Por esa razón hemos decidido utilizar una arquitectura orientada a servicios en la parte Back-End, y a componentes en la parte Front-End. En ambas casos, hemos seguido los siguientes principios (Martin, 2013):

- Principio Open/Closed. Este principio busca que los programas estén abiertos para la extensión y cerrados a la modificación. Es decir que no haga falta modificar el código de nuestras clases para extender su funcionalidad. Esto permite depurar de una manera más eficiente el código y ampliar su funcionalidad con otros métodos. Este principio lo podemos ver en nuestra aplicación en el uso de diferentes servicios web que aportan funcionalidad individualmente evitando el uso de un solo

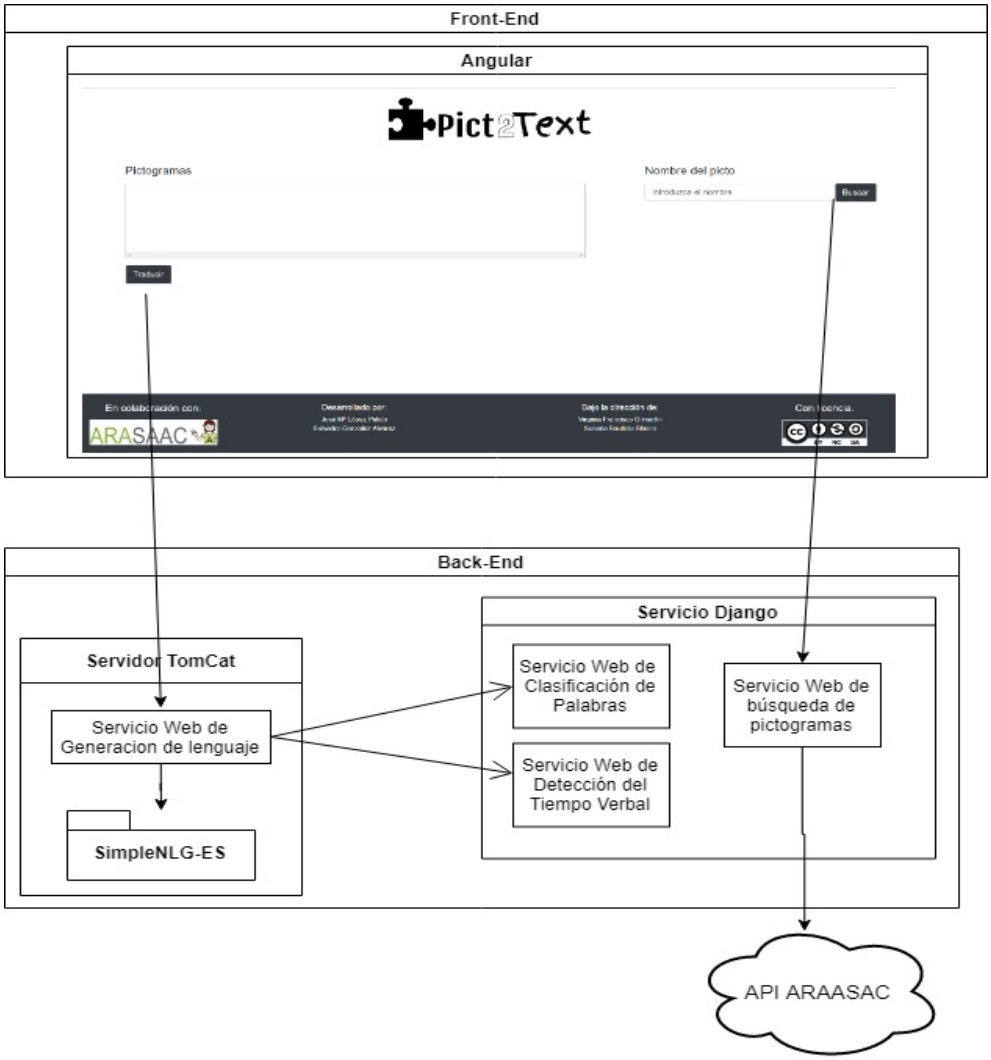


Figura 4.2: Visión general de Pict2Text

servicio web que implemente toda la funcionalidad. También se ve este principio en el proceso de generación de lenguaje donde las clases implementan pequeños métodos que encadenados aportan funcionalidad a la traducción.

- Principio de segregación de la interfaz. Este principio busca abstraer lo máximo posible las interfaces de todas las clases, para así poder reaprovecharlas en otras clases. Un ejemplo es como hemos implementado en el Front-End la parte de las llamadas a los servicios web del Back-End: todos los componentes de la aplicación hacen uso de la misma clase para lanzar las peticiones a los servicios web.

- Principio de inversión de dependencias. Consiste en evitar dependencias externas dentro del código, y que el código central de las aplicaciones no dependa de frameworks, de bases de datos o del como se conecten los diferentes servicios entre ellos. Normalmente se habla de que las clases de alto nivel no deben de depender de las clases de bajo nivel[2]. Un ejemplo de esto en nuestra aplicación es el uso de servicios web, que reciben tipos básicos para que estos no estén acoplados a estructuras de datos, puedan ser reutilizados por otras aplicaciones por otros desarrolladores.

Estos principios tienen como finalidad crear un producto software de calidad y robusto, gracias al desarrollo de un código ordenado y limpio, otorgando además una gran flexibilidad. Gracias a esto se consigue un producto reutilizable y fácil de mantener gracias al bajo acoplamiento que se obtiene.

En nuestra aplicación por tanto hay dos grandes partes diferenciadas tal y como se puede ver en la Figura 4.2 :

- Back-End: Conjunto de servicios web implementados en Java y Python que dotan de funcionalidad a la aplicación.
- Front-End: aplicación web basada en componentes usando Angular.

En las próximas secciones explicaremos en detalle cada una de estas partes.

4.2. Back-End

A lo largo de está sección presentaremos los diferentes servicios web desarrollados, los cuales hemos decidido agrupar dependiendo de la finalidad de los mismos en dos grupos: servicios web encargados del Procesamiento de Lenguaje Natural (PLN), donde están agrupados todos los servicios web dedicados a la generación y análisis de lenguaje y los servicios web para la gestión de pictogramas donde están englobados aquellos servicios que tratan con pictogramas.

4.2.1. Servicios web para Procesamiento de Lenguaje Natural

Dentro de está subsección hablaremos de los servicios web encargados del Procesamiento de Lenguaje Natural (PLN). Dentro de este grupo están englobados tres servicios web:

- Servicio Web de Clasificación de Palabras[3]: encargado de obtener la categoría léxica de un conjunto de palabras.

[2] https://devexperto.com/principio-de-inversión-de-dependencias/
[3] https://holstein.fdi.ucm.es/tfg-pict2text/translate/getWordAttrs

- Servicio Web de Detección del Tiempo Verbal[4]: encargado de detectar el tiempo verbal de una frase dado el conjunto de palabras que la componen.

- Servicio Web de Generación de Frases[5]: encargado de la generación de una frase en lenguaje natural a partir de un conjunto de palabras.

4.2.1.1. Servicio Web de Clasificación de Palabras

El servicio web de clasificación de palabras, se encarga de dada una palabra o un array de palabras devolver su categoría léxica (verbo, nombre, adverbio, determinante, adjetivo, pronombre, número o conjunción) y los atributos de cada una de las palabras recibidas, los atributos devueltos según el tipo de palabra se pueden ver en la Tabla 4.1.

Una vez se tiene la entrada se llama a la clase Tokenizer[6] de la librería Spacy. La salida de la clase Tokenizer sera procesada para generar el JSON de la salida de nuestro sistema. Para cada palabra ademas de la categoría léxica obtenemos los atributos mostrados en la Tabla 4.1, según la categoría léxica de la palabra los atributos obtenidos pueden variar. En la Figura 4.4, podemos ver un resumen del flujo que sigue este servicio.

Categoría Léxica	Atributos
Verbo	Forma verbal
Nombre	Género, número
Determinante	Género, número, definido o indefinido
Adjetivo	Género, número
Pronombre	Género, número, persona, definido, personal
Adverbio	-
Preposiciones	-
Conjunciones	-

Tabla 4.1: Tabla de atributos devueltos por el servicio según la categoría léxica de la palabra.

En la Figura 4.3, podemos ver la respuesta del servicio web para la entrada [corredor, despistado, perder, zapatillas, ayer].

[4] https://holstein.fdi.ucm.es/tfg-pict2text/translate/getTypePhrase
[5] https://holstein.fdi.ucm.es/tfg-pict2text/NLGWebService/createPhrase
[6] https://spacy.io/api/tokenizer

```
[
    {
        "attrs": {
            "Gender": "Masc",
            "Number": "Sing",
            "Type": "NOUN"
        },
        "keyword": "corredor"
    },
    {
        "attrs": {
            "Gender": "Masc",
            "Number": "Sing",
            "Type": "ADJ"
        },
        "keyword": "despistado"
    },
    {
        "attrs": {
            "Type": "VERB",
            "VerbForm": "Inf"
        },
        "keyword": "perder"
    },
    {
        "attrs": {
            "Gender": "Fem",
            "Number": "Plur",
            "Type": "NOUN"
        },
        "keyword": "zapatillas"
    },
    {
        "attrs": {
            "Type": "ADV"
        },
        "keyword": "ayer"
    }
]
```


Figura 4.3: Respuesta del servicio web de clasificación de palabras para la entrada [corredor, despistado, perder, zapatillas, ayer].

4.2.1.2. Servicio Web de Detección de Tiempo Verbal

Este servicio web, tiene como finalidad determinar el tiempo verbal de una frase dada la lista de palabras que conforman la frase. Este servicio busca las palabras temporales: ayer, hoy o mañana en el array de palabras

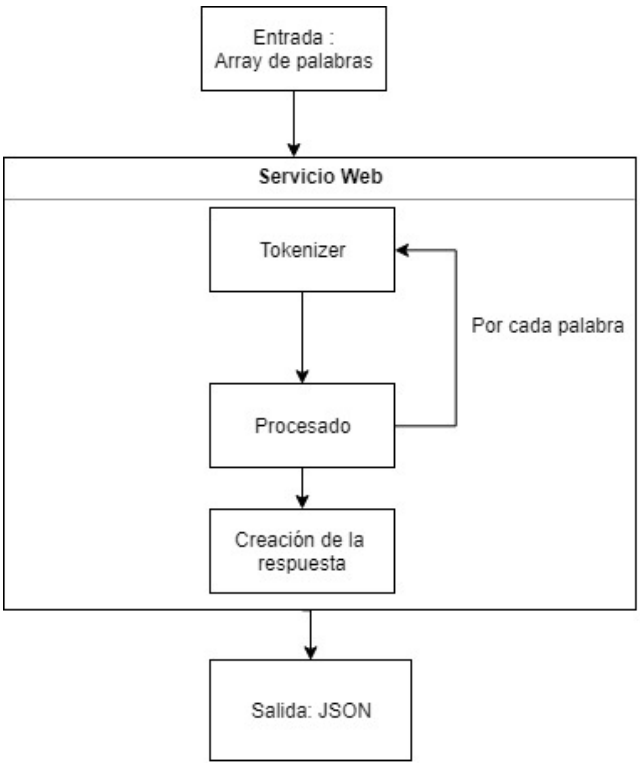


Figura 4.4: Flujo del servicio web de clasificación de palabras

recibido como entrada y si encuentra alguna devuelve el tiempo verbal que corresponda a la palabra temporal encontrada, pasado si encontró ayer, presente si encontró hoy y futuro si encontró mañana. En caso de no encontrar ninguna de las tres palabras devuelve presente por defecto. En la Figura 4.5, podemos ver un resumen del flujo que sigue este servicio y en la Figura 4.6 podemos ver un ejemplo de salida para la entrada [corredor, despistado, perder, zapatillas, ayer].

4.2.1.3. Servicio Web de Generación de Frases

Este servicio web recibe un array de palabras y devuelve una frase creada a partir de las palabras recibidas en la llamada. Los pasos seguidos por el servicio para crear la frase de salida son los siguiente:

1. Obtención de la categoría léxica de cada una de las palabras. Se llama al Servicio Web de Clasificación de Palabras explicado en la Sección 4.2.1.1 pasandole todo el array de palabras recibido como entrada. Así se obtienen las características y la categoría léxica de cada palabra.

2. División de palabras en sintagma nominal y en sintagma verbal. Una

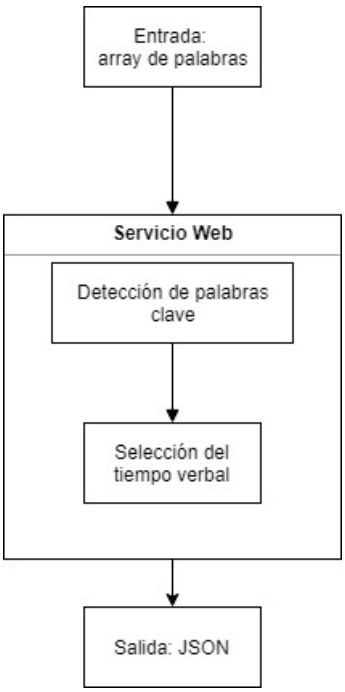


Figura 4.5: Flujo del servicio web de detección del tiempo verbal


```
{
    "Type": "past"
}
```


Figura 4.6: JSON de salida del servicio web de detección del tiempo verbal para la entrada [corredor, despistado, perder, zapatillas, ayer]

vez conocemos la categoría léxica de las palabras, buscamos el verbo dentro del array de palabras. Todas las palabras anteriores al verbo formaran el sintagma nominal y todas las palabras posteriores formaran el sintagma verbal.

3. Una vez identificadas las palabras que forman parte del sintagma nominal se pasa a crear esta parte de la frase para ello se hace lo siguiente:

   - Creamos el sintagma nominal que sera el objeto contenedor de todas las palabras que formaran el sujeto llamando a la función **createNounPhrase()** de la librería SimpleNLG.
   - Se establece el determinante del sujeto utilizando la función **setDeterminer()** de SimpleNLG sobre el objeto creado anteriormente. A esta función se le pasa el determinante o pronombre de

la lista de palabras de entrada. En caso de que la lista de palabras que forman el sujeto no contenga un pronombre o un determinante se añade el determinante por defecto "el", dado que SimpleNLG no añade ningún determinante por defecto.

- Se establece como palabra principal del sujeto la palabra con categoría léxica **NOUN** utilizando el método **setFeature()**. Además, se obtiene el género y número de la palabra principal del sujeto y usando las funciones **setFeature()** y **setPlural()** se establecen el género y el número del sintagma nominal. Al establecer el género y el número del sujeto todas las palabras que lo conforman utilizaran ese género y ese número. Por ejemplo si se recibe la entrada [mujeres, contento] el sujeto resultante sera "Las mujeres contentas".
- En caso de encontrar un adverbio dentro del array de palabras del sintagma nominal se fuerza su posición dentro del sintagma nominal al final. Por ejemplo, para la entrada [ayer, profesora, despistada, comer, galletas] se generara la frase "La profesora despistada ayer comió galletas"

4. Para crear el sintagma verbal se siguen los siguientes pasos:

   - Se genera un objeto que sera el sintagma verbal, con la función **createNounPhrase()**.
   - Se añade el determinante con una lógica parecida al sujeto, con la diferencia de que el determinante por defecto es "un", y solo se añadirá en caso de que exista una palabra con categoría léxica **NOUN**. Por ejemplo, si el array de entrada está formado por las palabras [niño, comer, galletas, duro] el resultado sera "El niño come unas galletas duras".
   - En caso de existir una palabra con categoría léxica **NOUN**, se aplicara su género y número al sintagma verbal, en caso de no existir se aplicara el género y número del sintagma nominal.
   - Si existe un adverbio se fuerza su posición al final de la frase, de manera que si las palabras de la entrada son: [profesora, comer, ayer, galletas, duro], la frase resultante sera "La profesora comió galletas duras ayer".

5. Se hace una llamada al Servicio Web de Detección de Tiempo Verbal explicado en la sección 4.2.1.2 con el array de palabras para obtener el tiempo verbal para conjugar la frase, y guardaremos el resultado para el ultimo paso.

6. Creamos un objeto frase con **SPhraseSpec()**. A continuación, mediante la función **.setSubject()** añadimos el sujeto creado previamente. En caso de tener verbo lo añadiremos utilizando **setVerbPhrase()** y fijaremos el tiempo verbal de la frase, con la función **setFeature()**.

7. Una vez tenemos todos la frase formada se pasa a la realización de la misma. Este proceso de realización utiliza el objeto **Realiser** de SimpleNLG para devolver un string que contendrá una frase construida utilizando los datos del punto anterior.

Entrada	Salida
Corredor, perder, zapatillas	El corredor pierde unas zapatillas
Corredor, rápido, perder, zapatillas	El corredor rápido pierde unas zapatillas
Corredor, ayer, perder, zapatillas	El corredor ayer perdió unas zapatillas.
Corredor, mañana, perder, zapatillas	El corredor mañana perderá unas zapatillas.
Corredor, rápido, ayer, perder, zapatillas	El corredor rápido ayer perdió unas zapatillas.
Corredor, perder, zapatillas, nuevo	El corredor pierde unas zapatillas nuevas
Corredor, rápido, perder, zapatillas, nuevo	El corredor rápido pierde unas zapatillas nuevas
Corredor, rápido, ayer, perder, zapatillas, nuevo	El corredor rápido ayer perdió unas zapatillas nuevas.
Corredor, rápido. perder, zapatillas, nuevo, ayer	El corredor rápido perdió unas zapatillas nuevas ayer.
Este, corredor, rápido, perder, zapatillas, nuevo	Este corredor rápido pierde unas zapatillas nuevas.
La, corredor, rápido, perder, zapatillas, nuevo	La corredora rápida pierde unas zapatillas nuevas.

Tabla 4.2: Tabla con diferentes entradas y salidas del Servicio Web de Generación de Frases en Lenguaje Natural.

En la Figura 4.7 podemos ver un resumen del flujo que sigue este servicio y en la Tabla 4.2 podemos varios ejemplos de entradas y salidas del servicio.

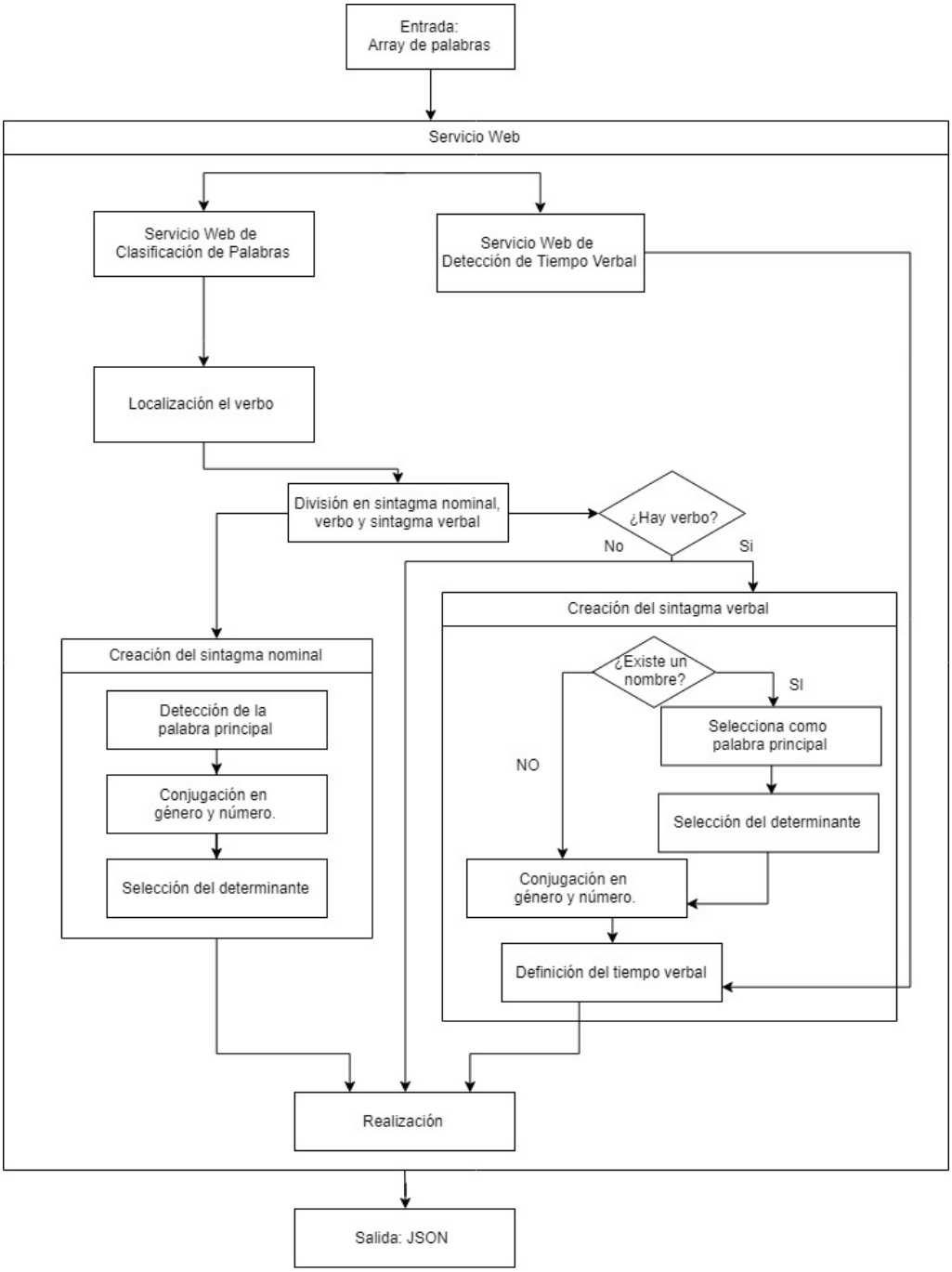


Figura 4.7: Flujo del Servicio Web de Detección de Generación de Frases en Lenguaje Natural

4.2.2. Servicios web para la Gestión de Pictogramas

En esta subsección pasaremos a explicar los dos servicios web que hemos creado para tratar con pictogramas:

- Servicio web para la obtención de pictograma dados una palabra[7].
- Servicio web para la busqueda de palabras asociadas a un pictograma dado su identificador[8].

Ambos servicios siguen la misma linea de ejecución, la cual se muestra en la Figura 4.8, primero leen la entrada, después hacen la llamada a la API de ARAASAC y dependiendo de si el resultado es un error o no, se creara una respuesta del servicio con el error o con el resultado del servicio.

4.2.2.1. Servicio Web de Búsqueda del Pictograma Asociado a una Palabra

Este servicio web, recibe una palabra a través de una petición GET y devuelve un objeto en formato JSON, que contiene un array con los diferentes pictogramas asociados a la palabra recibida. Este servicio lanza una petición GET con la palabra recibida al buscador de pictogramas de la API de ARAASAC[9], el cual devuelve una lista de todos los pictogramas asociados a la palabra introducida. Por cada pictograma nuestro servicio devuelve los siguientes atributos:

- Identificador del pictograma.
- Palabra asociada al pictograma.
- URL a la imagen del pictograma.

Cada pictograma con sus atributos se añade a una lista que sera devuelta en el resultado de este servicio web. En la Figura 4.9 podemos ver el resultado devuelto por el servicio para la entrada "cortar", esta palabra tiene varios pictogramas asociados, los cuales podemos ver en la Figura 4.10.

4.2.2.2. Servicio Web de Traducción de Pictogramas a Texto

Este servicio web recibe el identificador de un pictograma y devuelve las diferentes palabras asociadas al pictograma, ya que un solo pictograma puede tener varias palabras asociadas. En la Figura 4.11 podemos ver la

[7] `https://holstein.fdi.ucm.es/tfg-pict2text/picto/getPicto?pictoName=‘PalabraRelacionada’`

[8] `https://holstein.fdi.ucm.es/tfg-pict2text/picto/getPictoTranslate?pictoId=‘IdPictograma’`

[9] `https://api.arasaac.org/api/pictograms/api/pictograms/es/search/‘Nombre’`

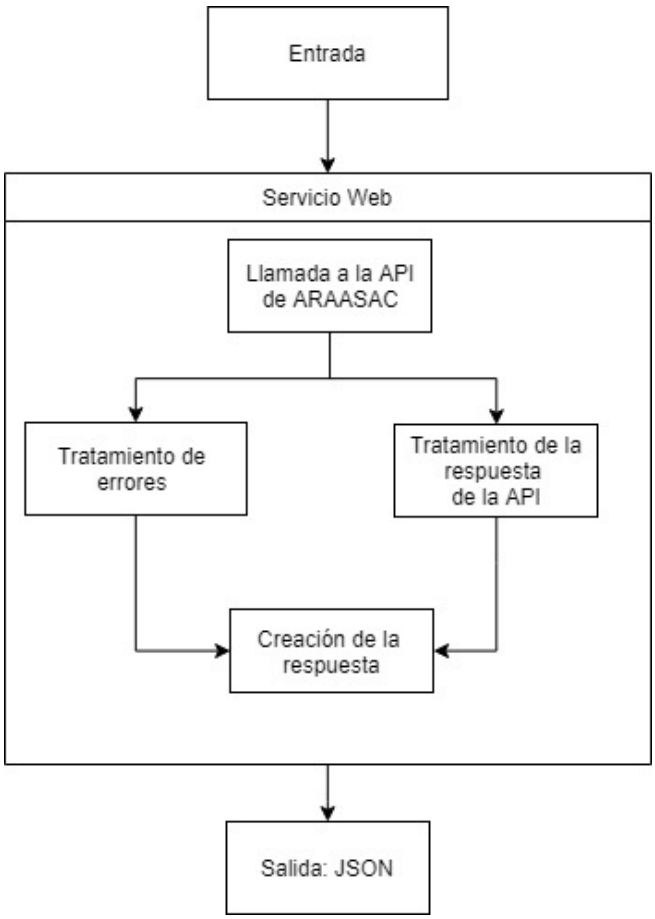


Figura 4.8: Flujo de los servicios web Para Pictogramas

respuesta para el identificador 7823, que tiene dos palabras asociadas: profesor y profesor de educación física. Este servicio lanza una petición con el identificador recibido, al buscador de pictogramas por id de la API de ARASAAC[10]. Después, en la respuesta de la API de ARAASAC se accede al atributo "Keywords" y el contenido de este atributo se encapsula dentro de un objeto que contiene un objecto con un array de significados, que sera la respuesta de este servicio tal y como se puede ver en la Figura 4.12.

4.3. Front-End

En esta sección introduciremos los diferentes servicios y componentes Angular desarrollados para la parte Front-End de la aplicación, la cual podemos ver en la Figura 4.13. Hemos implementado dos componentes:

[10]https://api.arasaac.org/api/pictograms/es/'idPictograma'

```
{
    "pictos": [
        {
            "id": 2547,
            "keyWords": "cortar",
            "url": "https://api.arasaac.org/api/pictograms/2547"
        },
        {
            "id": 2720,
            "keyWords": "cortar",
            "url": "https://api.arasaac.org/api/pictograms/2720"
        },
        {
            "id": 5975,
            "keyWords": "cortar",
            "url": "https://api.arasaac.org/api/pictograms/5975"
        },
        {
            "id": 25020,
            "keyWords": "cortar",
            "url": "https://api.arasaac.org/api/pictograms/25020"
        },
        {
            "id": 25022,
            "keyWords": "cortar",
            "url": "https://api.arasaac.org/api/pictograms/25022"
        }
    ]
}
```


Figura 4.9: Respuesta del servicio web de búsqueda del pictogramas asociados a una palabra, para la palabra cortar

- Buscador de pictogramas. Este componente sirve para buscar los pictogramas con los que se va a componer el mensaje que se desea traducir. El usuario introducirá el nombre para el que desea un pictograma pulsará buscar seleccionará uno de los resultados devueltos y pulsará en el botón de añadir para pasar el pictograma al espacio reservado para

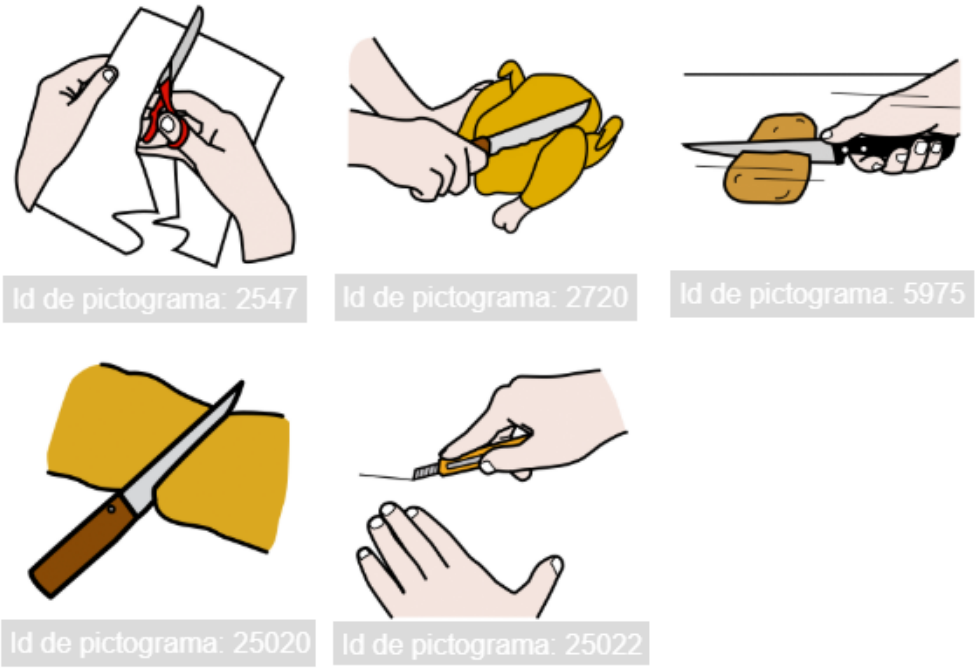


Figura 4.10: Pictogramas asociados a la palabra cortar

el mensaje a traducir.

- Traductor de pictogramas a lenguaje natural: Este componente es el encargado de manejar el mensaje con pictogramas que se va a traducir. Los pictogramas del mensaje se pueden ordenar o borrar. Una vez se tenga listo el mensaje con pictogramas que se desea traducir se pulsara el botón de Traducir y debajo aparecerá el mensaje en lenguaje natural traducido.

Cabe destacar que cada uno de estos componentes posee un HTML, un controlador y un servicio de aplicación propio. A parte de estos componentes hemos implementado una serie de útiles, como un archivo de constantes (para almacenar las diferentes direcciones de los servicios web), un servicio proxy que permite abstraer las peticiones HTTP, y un servicio de ventanas modales para el tratamiento de errores. Cada uno de estos servicios se explicara en detalle en las siguientes secciones. Ademas, hemos tenido en cuenta que la aplicación sea responsive y hemos diseñado la vista para tener una versión móvil la cual se puede ver en la Figura 4.14.

```
{
  "authors": [
    {
      "_id": "5ca1dce1b439f90001f959bf",
      "name": "Sergio Palao"
    }
  ],
  "license": "2",
  "downloads": 0,
  "tags": [],
  "legacyTags": [],
  "type": 1,
  "_id": "5ca1dd04b439f90001fa0862",
  "status": 1,
  "created": "2009-02-04T14:05:51.000Z",
  "lastUpdated": "2009-02-04T14:07:23.000Z",
  "idPictogram": 7823,
  "synsets": [
    "10713754-n"
  ],
  "keywords": [
    {
      "downloads": 0,
      "sinonyms": [],
      "idLocution": "2749.mp3",
      "keyword": "profesor",
      "type": "2",
      "meaning": "m. y f. Persona que se dedica a la enseñanza",
      "plural": "profesores",
      "idKeyword": 2749
    },
    {
      "downloads": 0,
      "sinonyms": [],
      "idLocution": "6816.mp3",
      "keyword": "profesor de educación física",
      "type": "2",
      "meaning": "Persona que enseña educación física o gimnasia.",
      "plural": "profesores de educación física",
      "idKeyword": 6816
    }
  ]
}
```


Figura 4.11: Respuesta del buscador por id de la API de ARAASAC, para el id 7823

4.3.1. Servicio Proxy

El servicio proxy de la aplicación sirve para abstraer las diferentes llamadas HTTP del resto de la aplicación. Este componente sirve de punto de acceso a los servicios web explicados en la Sección 4.2. Los métodos: **getBy-**

```
▼meanings: ["profesor", "profesor de educación física"]
    0: "profesor"
    1: "profesor de educación física"
```


Figura 4.12: Respuesta del servicio web de traducción de pictogramas a texto

Figura 4.13: Pict2Text

Name() y **postElement()** se encargan de emitir las peticiones HTTP, get y post respectivamente, esperar la respuesta del servicio web y devolverla al servicio de aplicación.

La ventaja principal de usar un servicio proxy es que abstrae la lógica de acceso del resto de la aplicación, de manera que si los datos pasaran a estar en una base de datos habría que cambiar mucho menos lógica y además permite la reutilización del código de las llamadas HTTP en toda la aplicación. Cuando el servicio de aplicación de algún componente hace uso de alguno de los métodos del servicio proxy, se convierte en un subscriptor

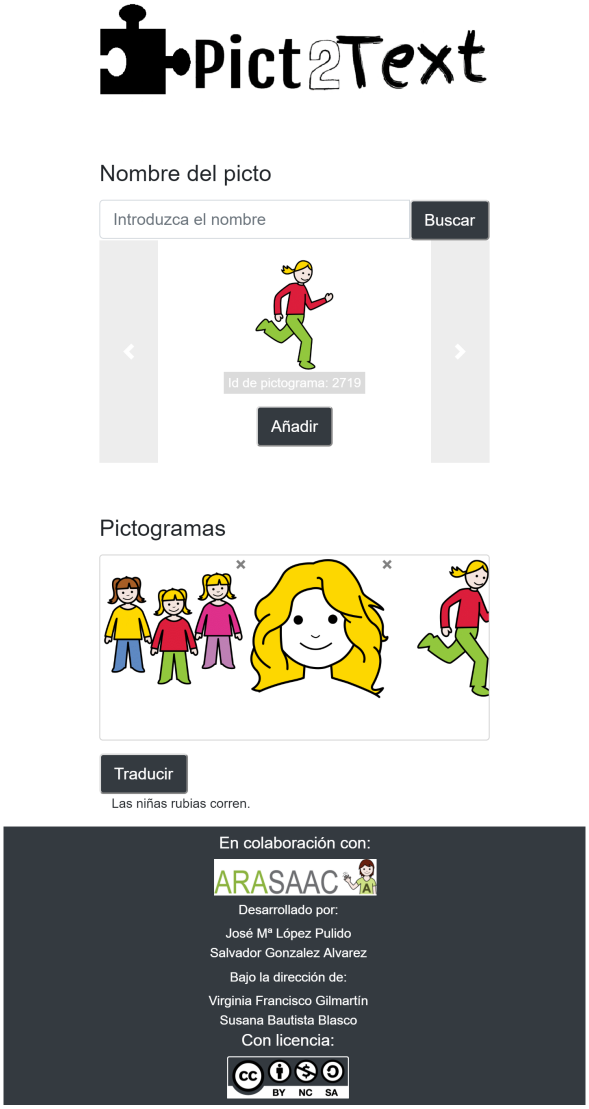


Figura 4.14: Versión de Pict2Text para dispositivos móviles

de dicho método. Gracias a la suscripción cuando el Servicio Proxy reciba la respuesta de la llamada HTTP, se encargara de avisar a sus suscriptores de que ya ha recibido la respuesta y entregarla. Esto es gracias al patrón Observer que implementa Angular en las llamadas HTTP. En caso de que la respuesta de la llamada fuera un error, el servicio proxy reintentará la llamada hasta cinco veces, si la ultima llamada falla, devuelve el error a los suscriptores.

En la Figura 4.15 podemos ver los métodos disponibles en la clase Proxy-

Service.

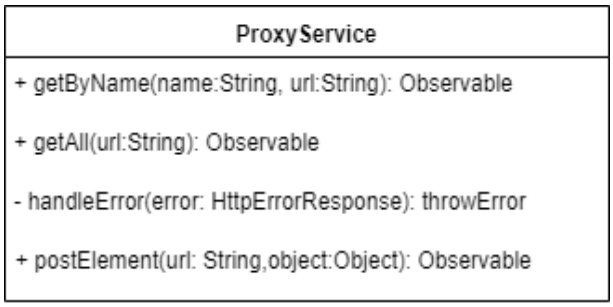


Figura 4.15: Diagrama de clase del Servicio Proxy

4.3.2. Servicio de Ventanas Modales

Este servicio tiene como finalidad mostrar los errores que hayan podido surgir a lo largo del flujo de la aplicación. El servicio de ventanas modales el cual hace uso todos los controladores de la aplicación Front-End, es un servicio reutilizable, que nos permite abstraer toda la lógica necesaria para mostrar los errores en un solo servicio. Este servicio tiene dos variables de entrada: el código de error y el texto asociado al error. Su utilidad principal es dado un error especifico construir y mostrar una ventana modal genérica para dar al usuario feedback de los errores. Este servicio nos permite tener un solo diseño de ventana modal para toda la aplicación.

El Servicio de Ventanas Modales sigue el patrón Factory, es decir, es capaz de crear objetos concretos, gracias a sus datos de entrada, utilizando una plantilla genérica, tal y como podemos ver en la Figura 4.16, donde podemos ver que lo único que cambia es el contenido de lo que muestra la ventana modal.

Además, hemos utilizado la librería NGBoostrap[11], esta librería hace uso del patrón Singleton para las ventanas modales y abstrae funcionalidad relacionada con el uso de ventanas modales, como por ejemplo la lógica de abrir o cerrar la ventana. Además de la funcionalidad tiene estilos CSS propios que ayudan a definir el tamaño de la ventana modal, el fondo que se pone sobre el body de la pagina, etc.

En la Figura 4.17, podemos ver un diagrama que resume los métodos de la clase ErrorModalService, y muestra su dependencia con la plantilla Error-Modal, la cual es usada para generar las ventanas modales.

[11] https://ng-bootstrap.github.io/

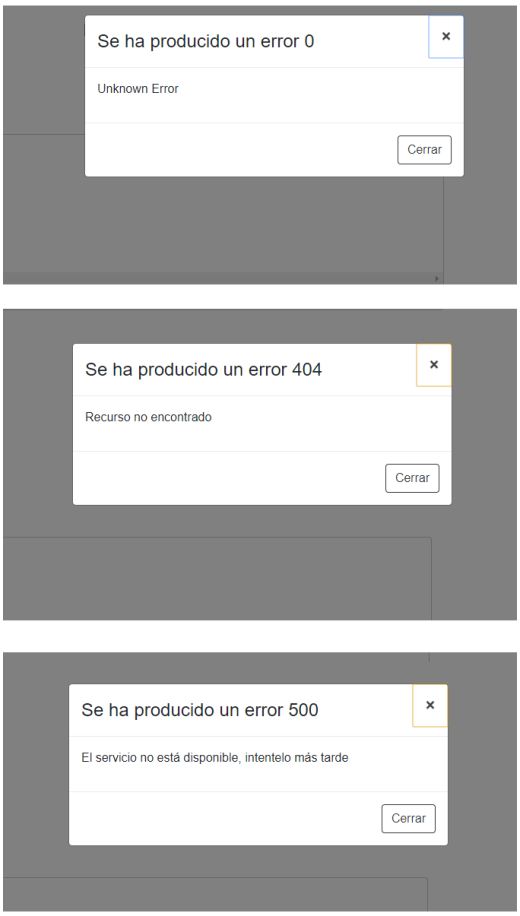


Figura 4.16: Ejemplo de diferentes ventanas modales

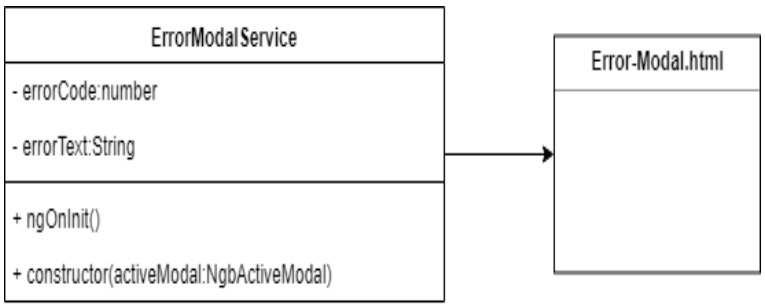


Figura 4.17: Diagrama del servicio de modales

4.3.3. Buscador de Pictogramas

La utilidad principal del buscador de pictogramas es ser el punto de entrada de los pictogramas a la aplicación. El componente esta formado por un

input donde introducir la palabra para la que se desea buscar el pictograma y un carrusel de pictogramas que nos permite mostrar todos los pictogramas asociados a la palabra introducida. El carrusel ha sido implementado con la librería NGBoostrap. Este componente posee un controlador, un servicio de aplicación y un transformer. El controlador es el encargado de conectar el HTML de la vista con el resto de la aplicación. Sirve para obtener la palabra para la que se desea buscar un pictograma y enviarlo al servicio de aplicación. Además de añadir los pictogramas al componente traductor de pictogramas usando el botón añadir. El servicio de aplicación creara la url para hacer la llamada al servicio web de búsqueda del pictogramas asociados a una palabra (visto en la Sección 4.2.2.1) usando el archivo de constantes. Con la url generada llamara al Servicio Proxy, que se encargara de enviar y recibir la respuesta del servicio web. Si la respuesta es correcta el servicio de aplicación hará uso del transformer para generar un objeto para el controlador. En caso de error devolverá el error al controlador para que muestre el mensaje de error. Podemos ver el diagrama del componente Buscador de Pictogramas en la Figura 4.18.

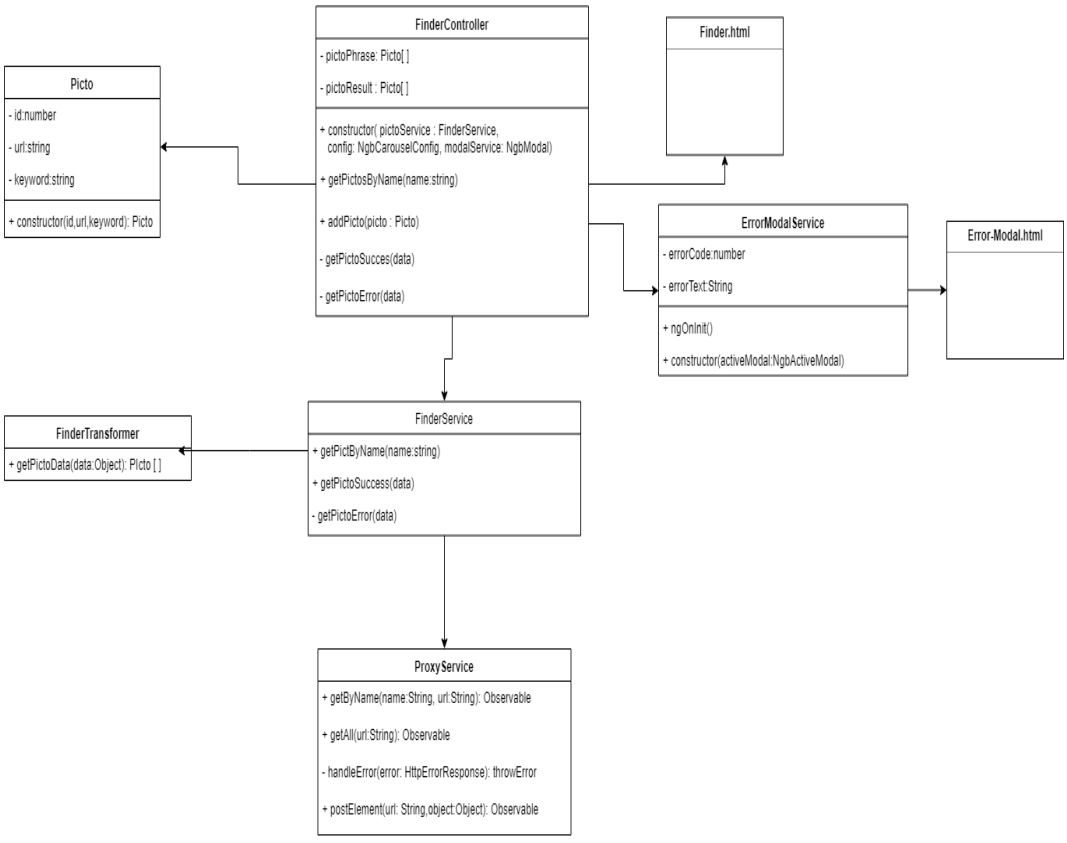


Figura 4.18: Diagrama del componente buscador de pictogramas

4.3.4. Traductor de Pictogramas a Lenguaje Natural

Este componente sirve de contenedor del array de pictogramas que después traduciremos y para lanzar la traducción. En el HTML el componente implementa un contenedor para los pictogramas, donde se muestran la frase con pictogramas, este contenedor se ha implementado usando la biblioteca Material[12] de Angular. Además, el componente tiene un botón para lanzar la traducción y la sección donde se mostrara el resultado de la traducción. Este componente tiene un controlador y un servicio de aplicación propio. El controlador se encarga de manejar la logica drag and drop que posee el contenedor de pictogramas, y la lógica de borrado de los pictogramas.

En la Figura 4.19, podemos ver un diagrama que muestra las clases que forman este componente así como los archivos HTML que contienen el código de la vista.

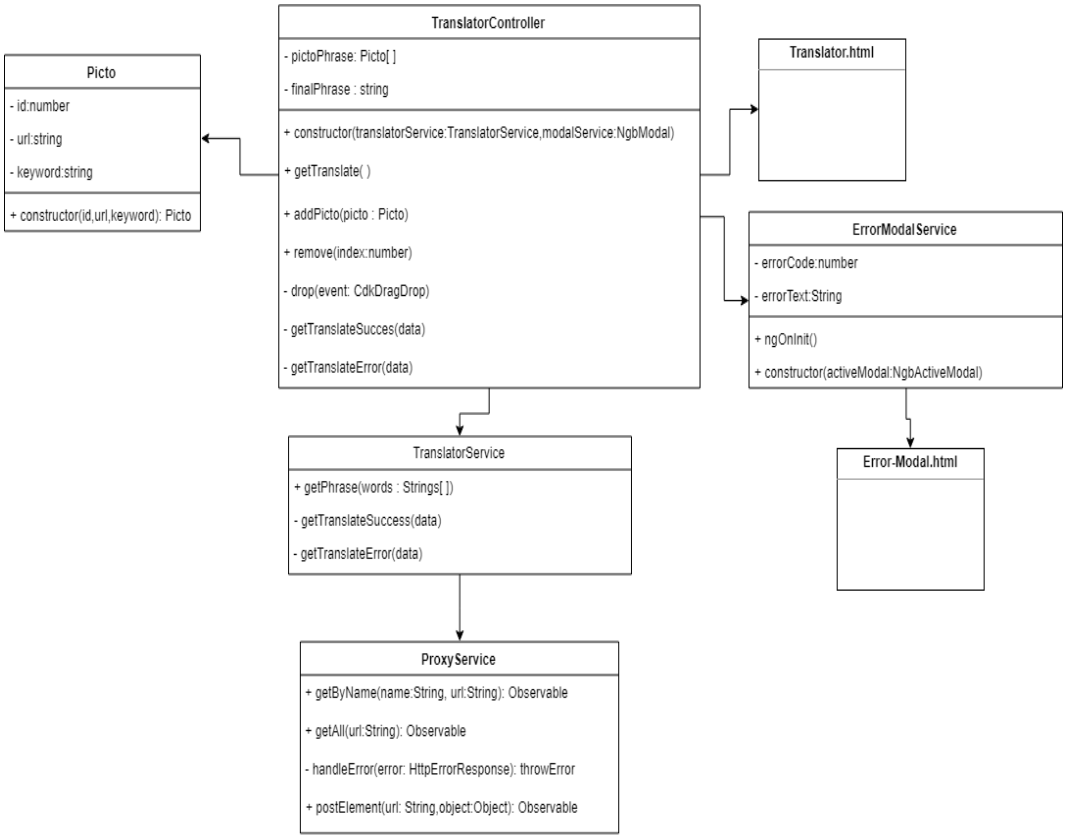


Figura 4.19: Diagrama del Traductor de Pictogramas a Lenguaje Natural

El servicio de aplicación tiene como parámetro de entrada el array de

[12]https://material.angular.io/

strings que formara la petición post, además obtiene la url del Servicio Web de Generación de Lenguaje del archivo de constantes y llamara al Servicio Proxy que sera el encargado de lanzar la petición. Una vez se obtiene la respuesta del Servicio Proxy se devuelve al controlador del componente, el cual ya sera el encargado de mostrar la traducción o mostrar la ventana de la modal en caso de error en la respuesta del servicio.

Capítulo 5

Evaluación

Cuando se finalizó la implementación de la aplicación web se realizó una evaluación de las traducciones realizadas por nuestra aplicación, para comprobar la cobertura y precisión de nuestro traductor. En este capítulo explicaremos el diseño de la evaluación en la Sección 5.1, los resultados obtenidos en dicha evaluación (Sección 5.2), y se hará un análisis de los resultados obtenidos en la Sección 5.3.

5.1. Diseño de la Evaluación

Para comenzar, es importante recordar que el público objetivo de nuestro traductor son aquellas personas que necesitan el apoyo de los pictogramas para comunicarse. Los pictogrmas permiten desde un nivel de comunicación muy básico (frases con pocos elementos, sin artículos, sin adverbios, sin complementos...) hasta un nivel de comunicación muy rico y avanzado (complementos, artículos, adverbios, preposiciones...) pero nunca tan completo y flexible como el que se puede lograr a través del lenguaje natural. En este trabajo hemos restringido el ámbito a la traducción de frases con pictogramas que van desde las frases más sencillas hasta frases algo más complejas con artículos, preposiciones, complementos... Por este motivo como corpus para la evaluación hemos utilizado corpus formados por frases, descartando cuentos, corpus con oraciones compuestas, etc.

Otra cuestión que se tuvo en cuenta a la hora de seleccionar los corpus para la evaluación fue el tipo de pictogramas utilizado. Nuestra aplicación está diseñada para trabajar con los pictogramas de ARASAAC por lo que los corpus para la evaluación debían utilizar este tipo de pictogramas. Y por eso, decidimos buscar en los corpus que ofrecía ARASAAC en su propia página web y así evitar lo máximo posible un problema de correlación entre los pictogramas del corpus y los que utiliza nuestra aplicación.

La evaluación se hizo de manera manual, introduciendo todos los ejemplos en nuestra aplicación y analizando caso por caso el resultado obtenido. Finalmente seleccionamos 4 corpus con 211 frases en total.

1. **Corpus de frases con fotos-pictogramas**[1]. Frases con pictogramas sencillos y sin nexos de unión, las frases solamente contienen las palabras principales de la oración. En total este corpus lo componen 34 frases. En la Figura 5.1 se pueden ver 3 frases de ejemplo del corpus. En el Apéndice A se puede consultar el corpus completo.

Figura 5.1: Frases del corpus fotos-pictogramas.

2. **Corpus de frases verdad o mentira**[2]. Este corpus está compuesto por frases sencillas junto a otras con un nivel más complejo, entre los que destacan frases con nexo de unión o frases con más de un sustantivo en el predicado. En total este corpus está formado por 115 frases. En el ejemplo de la Figura 5.2 podemos ver algunas de las frases contenidas en este corpus. En el Apéndice B se puede consultar el corpus completo.

[1] http://www.arasaac.org/materiales.php?id_material=474
[2] http://www.arasaac.org/materiales.php?id_material=691

Figura 5.2: Ejemplo del corpus verdad o mentira.

3. **Corpus de conciencia léxica modelos de frases**[3]. Frases con pictogramas sencillos, cada frase está formada por un máximo de 5 elementos, donde cada palabra de la frase tiene su pictograma para la traducción. En total este corpus está compuesto por 30 frases. En la Figura 5.3 se pueden ver 3 frases de ejemplo del corpus. En el Apéndice C se puede consultar el corpus completo.

Figura 5.3: Ejemplo del corpus conciencia léxica modelos de frases.

[3] http://www.arasaac.org/materiales.php?id_material=1253

4. **Corpus de frases absurdas**[4]. Este corpus está compuesto por frases simples junto con frases en las que aparece la negación del verbo, tal y como se puede observar en la Figura 5.4, donde aparecen 2 frases ejemplo del corpus. En total este corpus cuenta con 32 frases. En el Apéndice D se puede consultar el corpus completo.

Figura 5.4: Frases del corpus frases absurdas.

5.2. Resultados de la Evaluación

En la Tabla 5.1 se muestran los resultados de la evaluación de los 4 corpus: número total de frases procesadas, número de frases correctas y porcentaje de frases correctas.

[4] `http://www.arasaac.org/materiales.php?id_material=1701`

CORPUS	NºFRASES	FRASES CORRECTAS	% ACIERTO
Frases con fotos-pictos	34	14	41 %
Frases verdad o mentira	115	23	20 %
Conciencia léxica	30	21	70 %
Frases absurdas	32	17	53 %
Total	211	75	35 %

Tabla 5.1: Resultados de la evaluación.

Como podemos observar en la tabla en el corpus formado por frases sencillas y un mayor número de pictogramas en la traducción de cada frase (corpus de conciencia léxica modelos de frases) presenta un porcentaje de acierto del 70 %, los corpus que también cuentan con frases sencillas pero son más restrictivos en los pictogramas a traducir (corpus de fotos-pictogramas y corpus de frases absurdas) el porcentaje de acierto es de un 41 % y un 53 % respectivamente, mientras que en el otro corpus, al estar compuesto por frases con un nivel mayor de complejidad, el porcentaje de acierto es más bajo (20 %).

5.3. Análisis de los Resultados

Tras un análisis exhaustivo de los resultados de las frases traducidas incorrectamente, hemos detectado que los errores principales se deben a las siguientes causas:

TIPOS DE FALLOS	NºFALLOS/TOTAL FALLOS	% FALLO
Inferir preposiciones	49/139	35,25 %
Frases reflexivas	18/139	12,95 %
Más de un verbo en la oración	3/139	2,16 %
Más de un sustantivo en el sujeto	2/139	1,43 %
Pictogramas tachados	12/139	8,63 %
Errores analizador del lenguaje	12/139	8,63 %
Inferir determinante automáticamente	32/139	23,02 %
Colocar preposiciones lugar adecuado	11/139	7,91 %

Tabla 5.2: Frases traducidas incorrectamente por cada tipo de error.

En la Tabla 5.2 se presenta el número de frases traducidas incorrectamente debido a cada uno de los errores detectados.

- **Inferir preposiciones**. Nuestro sistema no está diseñado para añadir preposiciones de forma automática. Por ello, frases como la tercera frase mostrada en la Figura 5.2 ("el tren va **por la vía**"), es traducida por nuestro sistema como: "El tren va una vía".

- **Frases reflexivas**. Nuestra aplicación no contempla la traducción de frases reflexivas como: "la niña se lava las manos". En este caso Pict2Text devuelve "El niño lava unas manos".

- **Más de un verbo en la oración**. En esta primera etapa de desarrollo de la aplicación no se contempló la traducción de dos verbos en la misma oración, así que frases como: "Yo quiero ver la televisión" o "Yo quiero ir al baño", son traducidas por Pict2Text como: "Yo quiero un baño va" o "Yo quiero una televisión ve".

- **Más de un sustantivo en el sujeto**. De nuevo, Pict2Text no contempla la opción de tener más de un sustantivo en el sujeto. Cuando una frase contiene varios sustantivos en el sujeto como "El libro y el estuche están dentro de la mochila" o "La fresa y la pera son fruta" la traducción de Pict2Text devuelve "El estuche está una mochila dentro" o "La pera es una fruta".

- **Pictogramas tachados**. Como en los casos anteriores, Pict2Text actualmente no cuenta con la posibilidad de traducir pictogramas tachados que indiquen negación en la acción. Por tanto con frases como "El árbol no nada" o "La gallina no patina", pict2Text no es capaz de hacer esta traducción mediante el uso de un pictograma tachado, la única forma sería con el uso del pictograma "no" colocado delante del verbo.

- **Errores del analizador Spacy**. Spacy no tiene una precisión del 100 % tal y como se comentó en la Sección 3.3, así que la categoría léxica o los atributos devueltos por Spacy para una palabra pueden no ser correctos. Por este motivo la traducción realizada por nuestra aplicación falla en ejemplos como: "Mamá cose un **vestido** azul", ya que para Spacy vestido es un adjetivo y no un sustantivo y por ello nuestra aplicación traduce la frase como "La mamá cose vestida azul".

- **Inferir determinantes de forma automática**. Para la traducción de las frases se decidió que la aplicación añadiría automáticamente a los sustantivos en el sujeto el determinante el, la, los, las, concordando en género y número con el sustantivo. Si el sustantivo aparece en el predicado Pict2Text añade el determinante un, una, unos o unas, a no ser que aparezca en la frase un pictograma con el determinante explícito. Por ello en frases como: "Mamá conduce el coche", son traducidas por nuestra aplicación como "La mamá conduce un coche".

- **Colocar preposiciones lugar adecuado**. Actualmente la aplicación solo traduce las preposiciones si ponemos su pictograma explícitamente. Sin embargo, si después de la preposición viene un determinante acompañando a un sustantivo, la traducción da un resultado incorrecto, ya que nuestro generador traduce antes el determinante que la preposición. Es por ello que en frases como: "El niño juega con el oso" o "Los niños juegan con la pelota", son traducidas en Pict2Text por: "El niño juega el con oso" o "Los niños juegan la con pelota".

Los datos obtenidos de la evaluación han sido muy beneficiosos, ya que nos han ayudado a encontrar los tipos de traducciones a mejorar con el fin de obtener una aplicación más completa. Además, teniendo en cuenta que no se ha encontrado otra aplicación que haga traducciones parecidas a las que hace Pict2Text, podemos calificar estos resultados como bastante alentadores, ya que nos muestran que se ha realizado una aplicación con una buena base sólida aunque con mucho margen de mejora en el futuro.

Capítulo 6

Trabajo Individual

A lo largo de este capitulo hablaremos del trabajo realizado por cada uno de los miembros del equipo.

6.1. Trabajo Individual de Jose Maria

Lo primero que realicé fue un estudio de los SAACS, poniendo especial atención en los pictogramas, y de los servicios web. En la búsqueda de información sobre los SAACS, analicé los sistemas pictográficos más extendidos y la comunicación a través de pictogramas, además me puse en contacto con ARAASAC para obtener información sobre herramientas que nos permitieran usar los pictogramas de ARAASAC en nuestro trabajo, y así conseguimos acceso a su API.

Uno de los requisitos indispensables de este proyecto, era la necesidad de incluir la aplicación dentro de un servicio web, por lo cual investigué sobre cuáles eran las diferentes opciones para crear un Servicio Web en Python. Además desarrollé un prototipo tecnológico, para aprender a crear servicios web con Python.

A continuación, pensamos en la metodología de trabajo que seguiríamos y la documenté en la memoria, en la Sección 1.3.

El siguiente paso necesario, era un estudio sobre los diferentes sistemas de generación de lenguaje natural parte fundamental del proyecto. Investigué y documenté la base teórica de la generación de lenguaje natural y una vez terminé esto pase a la búsqueda de diferentes frameworks de generación de lenguaje natural en español, tras la cual encontré el framework SimpleNLG-ES. Una vez presentados los resultados de mi investigación en la memoria, desarrollé un prototipo tecnológico utilizando dicho framework. Este prototipo muy simple permitía crear una frase a partir de un sujeto, un verbo y un complemento directo. Para generar este servicio web, analicé diferentes

vías de servicios web en Java, y realice dos prototipos uno con WildFly[1] y otro con Apache Tomcat[2], y teniendo en cuenta diferente factores como rendimiento o la comunidad que había detrás, decidí implementar Tomcat.

Una vez tenía la base de la aplicación, desarrollé el servicio web de búsqueda de pictogramas asociados a una palabra y cree una pequeña aplicación para poder probar la integración de este servicio, con el servicio de generación de lenguaje. Una vez terminé el trabajo de implementación, documenté los servicios web, en la memoria.

Dado el límite del framework Django para la gestión de aplicaciones web desde el lado del cliente, desarrollé una aplicación web Front-End en Angular, que después integre con la aplicación Django, para de esta manera obtener mayor funcionalidad, flexibilidad y descargar parte del trabajo en el cliente. Una vez terminada la implementación documenté la aplicación y la arquitectura en la memoria.

Después realice un estudio sobre los diferentes analizadores de lenguaje natural para el castellano y decidimos usar Spacy. A continuación, desarrollé el servicio web de detección de palabras y posteriormente documenté su funcionamiento.

Además de este servicio decidimos que era necesario, crear un servicio web que detectara el tiempo verbal en el que se debería conjugar la frase, que también desarrollé.

Para el servicio de generación de lenguaje tuve que hacer una refactorización del servicio ya creado, ya que la complejidad del servicio debía ser mucho mayor. Gracias a esta refactorización conseguí un servicio web más fácil de mantener y de modificar. Una vez terminado todo el desarrollo documenté el servicio web y el flujo de su ejecución.

Para terminar con el desarrollo, desarrolle un diseño vista final que añadía a la aplicación que ya había desarrollado una cabecera y un footer, con el logo de la aplicación y el logotipo de ARAASAC y la licencia Creative Commons. Y hice junto con mi compañero el despliegue de la aplicación.

Lo último que realice fue el despliegue de la aplicación en el servidor[3] y realice algunas de las ultimas correcciones en la memoria y complete la documentación de las herramientas Angular y Django y el tema de trabajo futuro. Por ultimo, traduje la introducción y el resumen al inglés.

[1] http://www.wildfly.org/
[2] http://tomcat.apache.org/
[3] https://holstein.fdi.ucm.es/tfg-pict2text/

6.2. Trabajo Individual de Salvador

Cuando comenzamos a realizar este Trabajo Fin de Grado mi cometido principal junto con mi compañero José María fue realizar un estudio sobre los aspectos básicos en los que fundamentar las bases de nuestro proyecto. Entre estos aspectos destacan, saber que son los pictogramas, que función cumplen, público objetivo de nuestra aplicación, etc. A continuación realicé un estudio sobre los sistemas pictográficos existentes en la actualidad.

Una vez teníamos asentadas las bases sobre las que enfocaríamos nuestro trabajo tuve que realizar un análisis sobre que tipo de servicios web serían mejores para nuestro proyecto, entendiendo su funcionamiento y todo lo que nos aportaban, como esa independencia funcional entre cada uno de los servicios. Al mismo tiempo, empecé a investigar las diferentes herramientas que podríamos utilizar para implementar estos servicios web. Esta primera investigación nos llevo a la herramienta Django ya que preveíamos que podría ser una de las herramientas a utilizar en nuestra aplicación.

Cuando la investigación de mi compañero sobre Generación del Lenguaje Natural nos dio como resultado la idoneidad de SimpleNLG-ES para esta función, mi tarea se centró en realizar un estudio pormenorizado sobre el funcionamiento de dicho framework. Además de testear concienzudamente las aplicaciones basadas en pictogramas que existen actualmente.

El siguiente paso fue investigar el uso y funcionamiento de los servicios Apache Tomcat, los cuáles serían necesarios en nuestra aplicación.

Dada la limitación para la parte Front-End que presentaba el framework Django, documenté toda la información relevante sobre Angular, que fue la opción más idónea para ayudarnos a solventar dicha limitación.

Otro paso muy importante para la realización de nuestro trabajo fue elegir el analizador de lenguaje natural que mejor se adaptaba a nuestras necesidades, cuando observamos que la herramienta Spacy era la mejor opción, mi cometido fue documentar los motivos por los que dicha herramienta era la elección correcta y las características que nos ofrecía.

Una vez finalizada la implementación, me encargue de realizar un trabajo de mejora sobre la traducción que realizaba la aplicación, consiguiendo que aumentara su cobertura. Por ejemplo, añadiendo por defecto el pronombre "yo" en caso de no introducir ningún sujeto, añadiendo la traducción de más determinantes y pronombres, la posibilidad de traducir números, sustantivos en plural, etc.

Cuando se completó la fase de mejoras en la aplicación, fui el encargado de realizar la evaluación, escogiendo los corpus, realizando las traducciones de los ejemplos del corpus y realizando un análisis pormenorizado sobre el rendimiento y las posibles mejoras que se podrían realizar sobre ella.

Para ir completando la memoria, me encargué de redactar los apartados de agradecimientos y resumen, así como de escribir las conclusiones finales

del trabajo, añadir el trabajo futuro y realizar los apéndices con todos los ejemplos que se han estudiado en la evaluación.

Por último, realicé diferentes correcciones finales en la memoria y traduje el capítulo de conclusiones y trabajo futuro al inglés.

Capítulo 7

Conclusiones y Trabajo Futuro

A lo largo de este capítulo se presentarán las conclusiones principales de este TFG y el trabajo futuro que se podría realizar si se continuara el desarrollo del proyecto.

7.1. Conclusiones

En la actualidad la comunicación es un elemento indispensable para cualquier ser humano, ninguno de nosotros sería capaz de imaginarse su vida sin poder comunicarse con otras personas. Sin embargo, en nuestra sociedad hay personas con ciertas diversidades funcionales que les impiden poder tener una comunicación efectiva. Los sistemas aumentativos y alternativos de comunicación, como los pictogramas, ayudan a estas personas a mejorar su comunicación. A pesar de ello la comunicación con pictogramas continua siendo una comunicación reducida al entorno más cercano o a personas con el mismo tipo de diversidad. Esto es debido a que en la actualidad la comunicación con pictogramas no es universal ni comprensible para cualquier persona. Por todo ello, vimos necesario desarrollar una herramienta que facilitase la comunicación de las personas que necesitan los pictogramas para comunicarse con otras personas que no comprenden los pictogramas.

El objetivo principal en este TFG era desarrollar una aplicación web que permitiese traducir frases escritas con pictogramas a lenguaje natural con el fin de ayudar a aquellas personas que necesitan los pictogramas para comunicarse. Y para ello hemos desarrollado la aplicación web Pict2Text[1]. Pic2Text traduce correctamente oraciones verbales simples con sujeto, verbo y predicado tal y como se puede ver en la Figura 7.1. Si en el texto con pictogramas no aparecen explícitamente los artículos Pict2Text los añade haciendo la concordancia en género y número con el nombre al que acompañan como se

[1] `https://holstein.fdi.ucm.es/tfg-pict2text/`

observa en la Figura 7.1.

Figura 7.1: Ejemplo de traducción de la frase "El perro come unos macarrones".

Si por el contrario, sí aparecen estos elementos, Pic2Text también los admite y traduce correctamente la frase tal y como se puede ver en la Figura 7.2. Pict2Text también permite traducir frases que contengan adjetivos tanto en el sujeto como en el predicado tal y como se puede observar en la Figura 7.3.

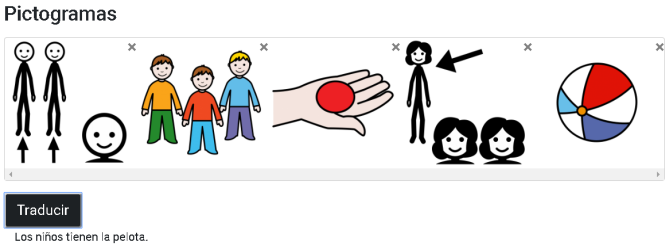


Figura 7.2: Ejemplo de la traducción de la frase "los niños tienen la pelota".

Además Pict2Text detecta automáticamente el tiempo verbal en el que se debe traducir la frase, tal y como se puede observar en la Figura 7.4, donde se conjuga la frase en pasado al encontrarse el pictograma de ayer.

Sin embargo aún queda mucho trabajo por hacer. Pict2Text no es capaz de inferir las preposiciones si no hay un pictograma con la preposición de forma explicita, como se puede ver en la Figura 7.5, donde habría que incluir la preposición "por" antes de "una vía". Tampoco hemos conseguido traducir oraciones reflexivas, oraciones con dos sustantivos en el sujeto, o con dos verbos, oraciones copulativas...

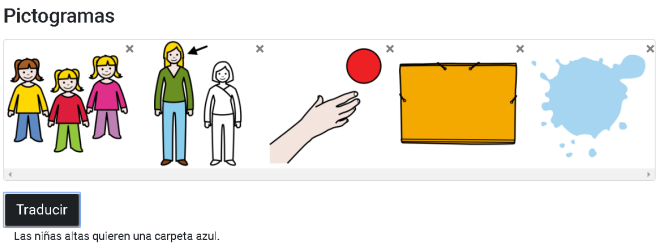


Figura 7.3: Ejemplo de la traducción de la frase “las niñas altas quieren una carpeta azul”.

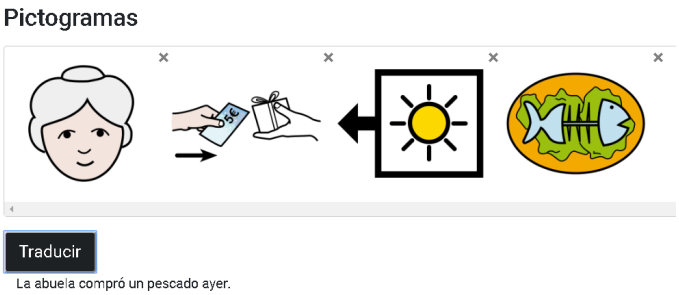


Figura 7.4: Ejemplo de la traducción de la frase “la abuela compró un pescado ayer”.

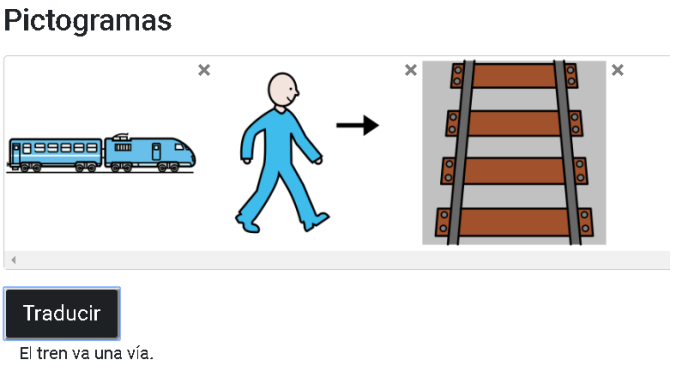


Figura 7.5: Ejemplo de la traducción de la frase “El tren va por una vía”.

Cabe destacar que al comienzo de este TFG no encontramos ninguna aplicación que realizara traducciones de pictogramas a texto así que podemos afirmar que aún sin ser perfecto el resultado de este trabajo establece unas bases solidas para la traducción de texto con pictogramas a lenguaje natural.

Otro de los objetivos era que la aplicación desarrollada fuera lo más cómoda, intuitiva y accesible posible a fin de poder llegar a la mayor cantidad de personas. La aplicación no ha sido evaluada en este sentido. Para poder determinar la usabilidad de nuestra aplicación deberíamos haber realizado una evaluación con usuarios finales. Aun así, creemos que es una aplicación bastante sencilla e intuitiva, aunque el hecho de que la única forma de introducir el mensaje de entrada con pictogramas sea creándolo desde cero con la ayuda del buscador, limita el número de personas que pueden usar la aplicación y su flexibilidad.

Otro objetivo de este trabajo era implementar las funcionalidades de la aplicación usando servicios web para que se pudieran usar en otras aplicaciones. Esto se ha conseguido desarrollando los servicios web utilizando como entrada tipos de datos básicos para que puedan ser usados por otro desarrolladores. Además, hemos desarrollado una aplicación Front-End basada en componentes de Angular, que son también reutilizables en otras aplicaciones, que utilicen Angular como framework.

Este proyecto, además, nos ha permitido aplicar muchos de los conocimientos adquiridos en las asignaturas cursadas durante el grado a un proyecto grande y con una utilidad más allá del ámbito académico. Entre las asignaturas que más nos han ayudado en este TFG cabe destacar las siguientes:

- **Ingeniería del Software**, **Modelado del Software** y **Gestión de Proyectos Software**. Donde nos enseñaron a tener la capacidad de gestionar proyectos software de una manera correcta y eficaz. Adoptando algunas de las características propias de las metodologías ágiles, además de una metodología adaptativa a las necesidades del trabajo. Además del uso de diferentes patrones y principios de arquitectura software.

- **Tecnología de la programación** y **Gestión de Información en la Web**. Donde nos enseñaron dos de los principales lenguajes utilizados en este TFG: Java y Python.

- **Aplicaciones Web**. En esta asignatura nos enseñaron los conocimientos sobre HTML, CSS y Javascript usados en el desarrollo de nuestra aplicación web.

- **Prácticas en empresa**. En la empresa donde realizó las practicas uno de los integrantes se utilizaba Angular, por lo que lo aprendido durante sus prácticas ha resultado muy útil en este proyecto.

- **Ética, Legislación y Profesión**. En esta asignatura nos informaron sobre el uso de las licencias, con las que proteger nuestro trabajo así como aprender a utilizar de una manera correcta el software libre en nuestro trabajo y así respetar la licencia de la API de ARASAAC.
- **Redes**. Donde obtuvimos algunos de los conocimientos básicos que hemos necesitado para el manejo de servicios web y la conexión entre los mismos.

Además este TFG también nos ha aportado muchos conocimientos nuevos, entre los que destacan: creación de servicios web en diferentes lenguajes de programación, despliegue de aplicaciones web, el uso de Latex, y un acercamiento a herramientas de procesamiento de lenguaje natural como son Spacy y SimpleNLG-ES.

7.2. Trabajo Futuro

Con el objetivo de mejorar y completar algunas funcionalidades de cara a obtener una aplicación más completa y que ofrezca una mayor cobertura, pensamos que se podrían marcar como trabajo futuro lo siguiente:

- **Reconocimiento de pictogramas**. Implementar otras alternativas en la aplicación aparte del buscador de pictogramas, para crear mensajes con pictogramas. Una nueva alternativa podría ser: la posibilidad de cargar directamente en nuestra aplicación un mensaje con pictogramas desde un archivo o poder hacer una foto a un mensaje escrito con pictogramas.
- **Evaluación de la usabilidad con expertos**. Realizar un diagnostico sobre si la aplicación creada es simple, intuitiva y útil para usuario con diversidad funcional que utilicen pictogramas, para ello realizaríamos un estudio colaborativo con expertos que nos puedan orientar y ver posibles mejoras.
- **Introducción de nuevos elementos en la traducción**. Actualmente nuestra aplicación solo contempla la traducción automática de determinantes. Habría que añadir la posibilidad de generar automáticamente otros elementos que no aparezcan explicitamente en el mensaje con pictogramas como pueden ser: preposiciones, conjunciones... Frases como la mostrada en la Figura 7.6 son traducidas por nuestra aplicación como “El tren va una vía”. Deberíamos buscar la forma de diferenciar cuando un sustantivo necesita ir acompañado de una preposición, por ejemplo.
- **Traducción de frases reflexivas.** Habría que detectar si hace falta poner un pronombre reflexivo en la oración. Para ello podríamos crear

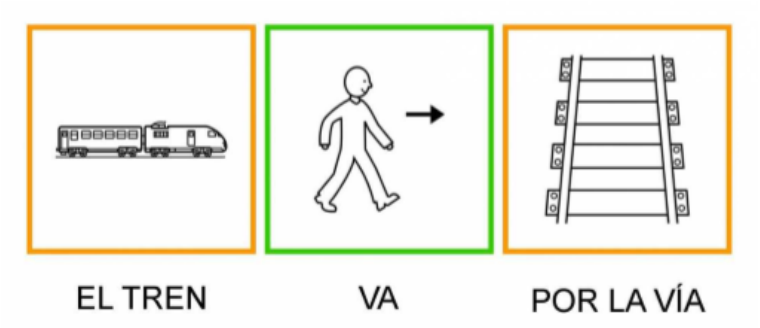


Figura 7.6: Ejemplo de la frase "El tren va por la vía".

un servicio web que reconozca aquellos verbos que son reflexivos y añadir los mecanismos necesarios para generar este tipo de frases.

- **Traducción de varios verbos**. Cuando haya dos pictogramas de verbos habría que estudiar si los dos verbos se encuentran en una posición consecutiva en la frase, de ser así, el primer verbo sería el que se conjugaría en presente, pasado o futuro y el segundo iría en infinitivo. Si por el contrario, vienen por separado habría que observar cual es el verbo principal de la frase con el fin de encontrar su sujeto, verbo y predicado, y después realizar la traducción pertinente. Así se traducirían frases como: "Yo quiero ver la televisión" y "Yo quiero ir al baño".

- **Traducción de varios sustantivos en el sujeto**. En caso de recibir varios pictogramas de sustantivos en el sujeto habría que clasificar ambos sustantivos como parte principal del sujeto para que ninguno pudiera borrar la información del otro y una vez hecho esto asignarles sus determinantes. Con esta implementación se traducirían frases del tipo: "El libro y el estuche están dentro de la mochila" o "La fresa y la pera son fruta".

- **Traducción de pictogramas tachados que indiquen negación.** Pict2Text no está preparada para recibir pictogramas tachados que indican negación. Para incorporar este tipo de pictogramas en nuestra aplicación se debería añadir la posibilidad de tachar un pictograma al componer el mensaje, y luego traducir la frase con ese pictograma tachado. Así se traducirían frases como: "El árbol no nada" o "La gallina no patina".

- **Inferir determinantes de manera automática**. Cuando el determinante no aparece explicitamente en el mensaje con pictogramas nuestra aplicación siempre añade como determinante automático el, la, los, las si el sustantivo está en el sujeto y un, una, unos, unas si está en el

predicado. Esto no siempre es correcto, para añadir un determinante más adecuado según el caso habría que obtener si el sustantivo es contable o incontable, y así acompañar a los sustantivos contables con los determinantes un, una..., y a los incontables con el, la...

- **Colocar preposiciones en el lugar adecuado**. La aplicación solo traduce preposiciones si ponemos su pictograma explicitamente en el texto, sin embargo si después de esa preposición viene un determinante, la traducción que hace queda de la siguiente forma "Los niños juegan la con pelota". Para arreglar este problema habría que estudiar previamente si en la frase a traducir hay una preposición, si es así, se traduciría toda la frase como un complemento consiguiendo mantener el orden correcto de la traducción. Así se lograría traducir correctamente frases como: "Los niños juegan con la pelota" o "El bebé juega con los cubos".

- **Ampliar la cobertura del servicio de detección del tiempo verbal de la frase**. Se deberían añadir nuevas palabras o expresiones para el reconocimiento del tiempo verbal de la frase, como por ejemplo los adverbios antes o después...

- **Reconocimiento de pictogramas que representan expresiones.** Algunos pictogramas como el pictograma mostrado en la Figura 7.7 que significa "Pasear al perro" necesitan de un procesamiento especial que separe la expresión en sus diferentes palabras para poder ser procesadas una a una por nuestro traductor.

Figura 7.7: Pictograma de "Pasear al perro".

Chapter 7

Conclusions and Future Work

Throughout this chapter will be presented the main conclusions of this work and the future work that can be done if the development of the project is continued.

7.1. Conclusions

Nowadays, communication is an essential element for anyone, none of us would be able to imagine his life without without communication with other people. However, in our society there are people with certain functional diversities that prevent them from having an effective communication. Augmentative and alternative communication systems, such as pictograms, help these people improve their communication. In spite of this, communication with pictograms continues to be a reduced communication to the closest environment even the people with the same type of diversity. This is because currently communication with pictograms is not universal or understandable to anyone. For all these reasons, we saw the need to develop a tool that facilitates the communication of the people who need the pictograms to communicate with other people who do not understand the pictograms.

The main objective in this work was to develop a web application that would allow to translate sentences written with pictograms into natural language in order to help those people who need the pictograms to communicate. And for this we have developed the web application Pict2Text[1]. Pic2Text translates correctly simple verbal sentences with subject, verb and predicate as can be seen in picture 7.1. If the text with pictograms the Pict2Text articles are not explicitly added, they are added by making the agreement in gender and number with the name that they are accompanying. It shows in Picture 7.1.

[1] `https://holstein.fdi.ucm.es/tfg-pict2text/`

Figure 7.1: Translation example of the phrase “El perro come unos macarrones”.

On the other hand, if these elements do appear, Pic2Text also supports them and correctly translates the phrase, as can be seen in Picture 7.2. Pict2Text also allows you to translate phrases that contain adjectives in the subject and the predicate, as can be seen in Picture 7.3.

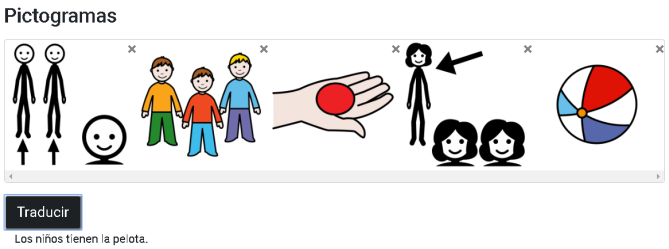


Figure 7.2: Translation example of the phrase “los niños tienen la pelota”.

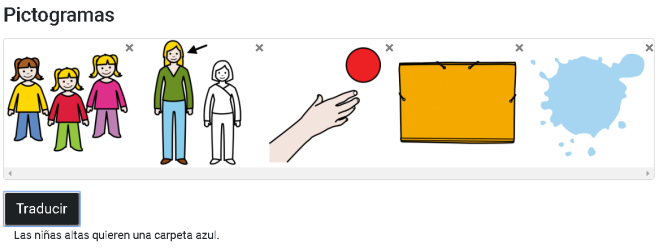


Figure 7.3: Translation example of the phrase “las niñas altas quieren una carpeta azul”.

In addition Pict2Text automatically detects the verb tense in which the phrase must be translated, as can be seen in Picture 7.4, where the phrase is conjugated in the past when the pictogram of yesterday is found.

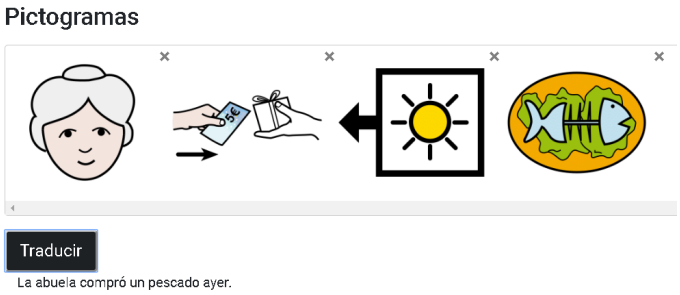


Figure 7.4: Translation example of the phrase "la abuela compró un pescado ayer".

However, there is still a lot of work to be done. Pict2Text is not able to infer prepositions if there is no pictogram with the preposition explicit shape, as can be seen in Picture 7.5, where the preposition should be included "por" before "una vía". Neither have we been able to translate reflective sentences,

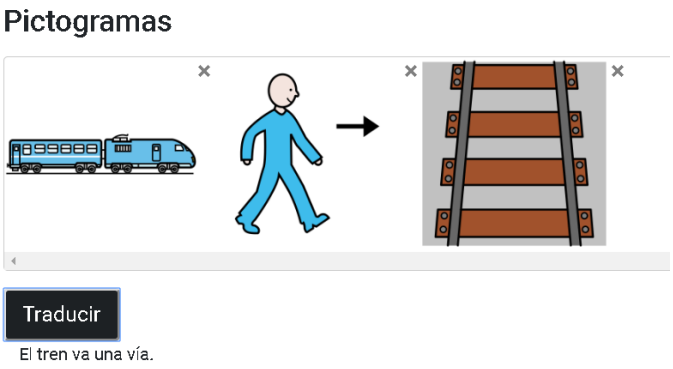


Figure 7.5: Translation example of the phrase "El tren va por una vía".

sentences with two nouns in the subject, or with two verbs, copulative sentences ...

It should be noted that at the beginning of this work we did not find any application that made translations of pictograms into text, so we can affirm that even without being perfect, the result of this work establishes solid bases for the translation of text with Pictograms to natural language.

Another objective was that the application developed was as comfortable, intuitive and accessible as possible in order to reach the largest number of people. The application has not been evaluated in this sense. In order to determine the usability of our application we should have done an evaluation with end users. Still, we believe it is a fairly simple and intuitive application, although the fact that the only way to enter the input message with pictograms is to creating From scratch with the help of the search engine, limits the number of people who can use the application and its flexibility.

Another objective of this work was to implement the functionalities of the application using web services so that they could be used in other applications. This has been achieved by developing web services using as input basic data types as so that they can be used by other developers. In addition, we have developed a Front-End application based on Angular components, which are also reusable in other applications, using Angular as a framework.

This project has also allowed us to apply many of the knowledge acquired in the subjects studied during the degree to a large project and with a utility beyond the academic field. Among the subjects that have helped us the most in this GFR it is worth highlighting the following:

- **Software Engineering**, **Modelado software** and **Software Project Management**. Where we were taught to have the ability to manage software projects in a correct and effective way. Adopting some of the characteristics of agile methodologies, as well as an adaptive methodology to the needs of the work. In addition to the use of different patterns and principles of software architecture.
- **Programming Technology** and **Web Information Management**. Where we were taught two of the main languages used in this work: Java and Python.
- **Web aplications**. In this subject we were taught the knowledge about HTML, CSS and Javascript used in the development of our web application.
- **Business practices**. In the company where he performed the practices one of the members was used angular, so that what learned during his practices has proved very useful in this project.
- **Ethics, Legislation and Profession**. In this subject we were informed about the use of licenses, with which to protect our work as well as learn to use the free software in a correct way in our work and thus respect the license of the API ARASAAC.

- **Networks**. Where we obtained some of the basic knowledge that we have needed for the management of Web services and the connection between them.

Furthermore, this work has also provided us with a lot of new knowledge, including: Creation of Web services in different programming languages, Web application deployment, Latex, and an approach to processing tools of Natural language as they are Spacy and SimpleNLG-ES.

7.2. Future Work

With the aim of improving and completing some functionalities in order to obtain a more complete application and offering greater coverage, we think that the following could be marked as future work:

- **Pictograms Recognition**. Implement other alternatives in the application other than the Pictograms Finder, to create messages with pictograms. A new alternative could be: the possibility to directly load in our application a message with pictograms from a file or to make a photo to a message written with pictograms.

- **Usability evaluation with experts**. Perform a diagnosis on whether the application created is simple, intuitive and useful for user with functional diversity that use pictograms, for this we would perform a collaborative study with experts who can guide us and see possible improvements.

- **Introduction of new elements in the translation**. Currently our application only contemplates the automatic translation of determinants. You would have to add the possibility of automatically generating other elements that do not appear explicitly in the message with pictograms such as: prepositions, conjunctions... Phrases like the one shown in the Picture 7.6 are translated by our application as "El tren va una vía". We should look for ways to differentiate when a noun needs to be accompanied by a preposition, for example.

- **Translation of reflective phrases.** It should be detected if it is necessary to put a reflective pronoun in the sentence. To do this we could create a Web service that recognizes those verbs that are reflective and add the necessary mechanisms to generate this type of phrases.

- **Translation of several verbs**. When there are two pictograms of verbs should be studied if the two verbs are in a consecutive position in the sentence, in that case, the first verb would be the one that would conjugate in present, past or future and the second would go

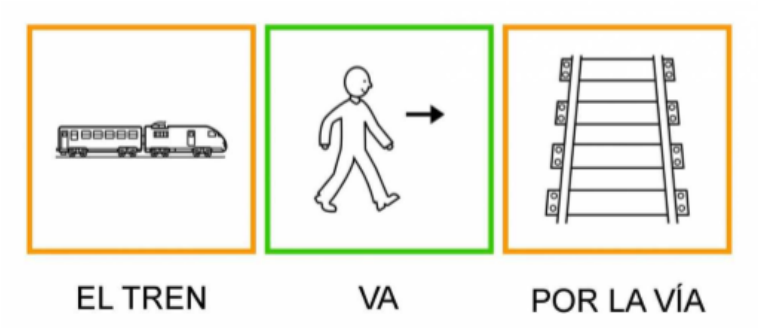


Figure 7.6: Translation example of the phrase "El tren va por la vía".

in infinitive. In contrast, they come separately would have to observe which is the principal verb of the sentence in order to find its subject, verb and predicate, and then to carry out the pertinent translation. This would translate phrases like: "Yo quiero ver la televisión" and "Yo quiero ir al baño".

- **Translation of various nouns in the subject**. In case of receiving several pictograms of nouns in the subject it would be necessary to classify both nouns as main part of the subject so that none could erase the information, and then, assigning their determinants. With this implementation, phrases of the type would be translated: "El libro y el estuche están dentro de la mochila" or "La fresa y la pera son fruta".

- **Translation of the cross out pictograms that indicate denial.** Pict2Text is not ready to receive cross out pictograms that indicate denial. To incorporate this type of pictograms in our application you should add the possibility to cross a pictogram when composing the message, and then translate the phrase with that pictogram crossed out. Thus would be translated phrases like: "El árbol no nada" o "La gallina no patina".

- **Deduce determinants automatically**. When the determinant does not appear explicitly in the message with pictograms our application always adds as an automatic determinant "el, la, los, las", if the noun is in the subject and "un, una, unos, unas", if it is in the predicate. This is not always correct, to add a more appropriate determinant as the case should be obtained if the noun is countable or uncountable, and thus accompanying the nouns with the determinants "un, una...", and the uncountables with "el, la..."

- **Place prepositions in the right place**. The application only trans-

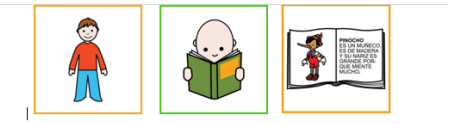
lates prepositions if we put its pictogram explicitly in the text, however if after that preposition comes a determinant, the translation that makes is in the following way “Los niños juegan la con pelota”. To fix this problem would have to study previously if the sentence to translate there is a preposition, if so, would translate the whole sentence as a complement getting the correct order of translation. This would make it appropriate to translate phrases like: “Los niños juegan con la pelota” o “El bebé juega con los cubos”.

- **Extend the verbal time detection service coverage of the phrase**. New words or expressions should be added for the recognition of the verbal tense of the phrase, for example, adverbs before or after...
- **Recognition of images representing expressions.** Some pictograms such as the pictogram shown in the Picture 7.7 which means “Pasear al perro” need a special processing that separates the expression in its different words in order to be processed one by one by our translator .

Figure 7.7: Pictogram “Pasear al perro”.

Apéndice A

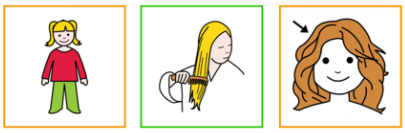
Corpus: frases con fotos y pictogramas



El niño lee un cuento.

La niña juega con el agua.

La niña se cepilla los dientes.



La niña se peina el pelo.

Mamá conduce el coche.

Mamá cepilla al perro.

El perro juega con el hueso.

La niña se columpia en el columpio.

Las niñas bailan ballet.

La niña monta en el triciclo.

Los niños juegan con la nieve.

Mamá corta el tomate.

El niño se disfraza de oso.

Los niños juegan con las construcciones.

La niña hace los deberes.

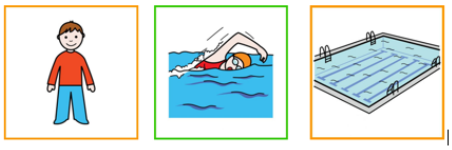
Los niños juegan al futbol.

La niña monta en bicicleta.

La niña juega con las piedras.

Mamá lee con la niña.

La niña lleva un paraguas.



El niño nada en la piscina.

Los niños hacen bolas de nieve.

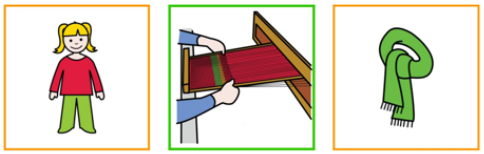
Los niños juegan en el parque.

El niño pela la manzana.

La niña pinta un dibujo.

Las niñas saltan a la comba.

Las niñas de mojan con agua.

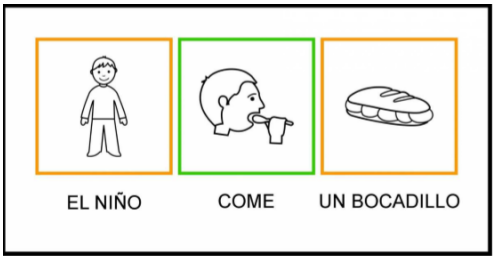


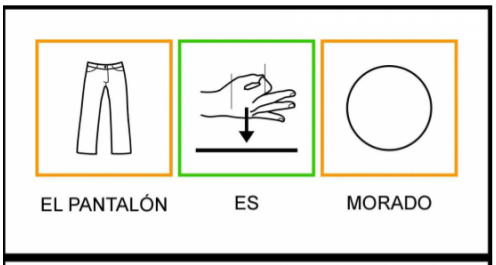
La niña teje una bufanda.

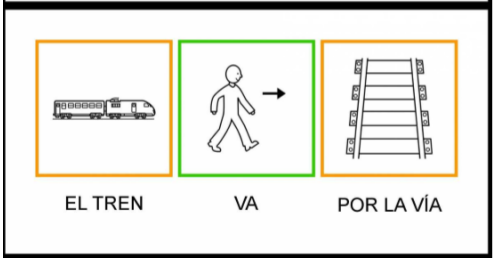
Los niños tocan las guitarras.

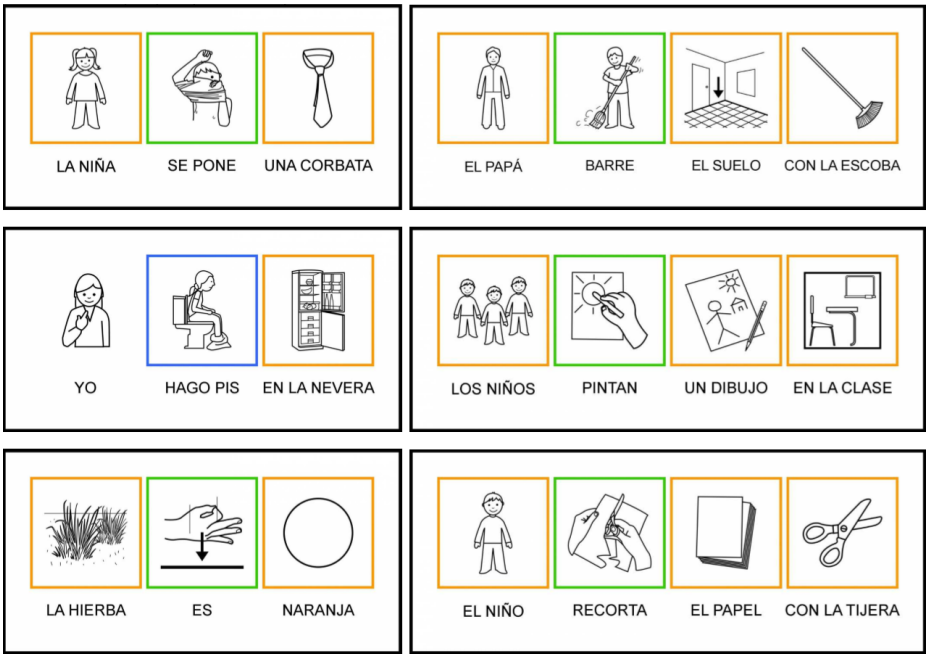
Apéndice B

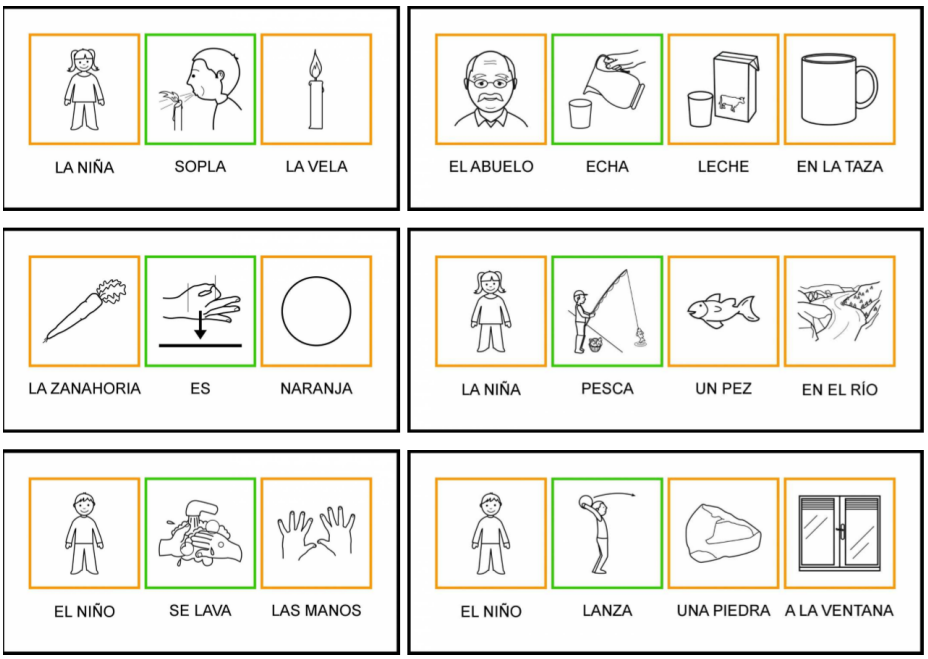
Corpus: frases verdad o mentira

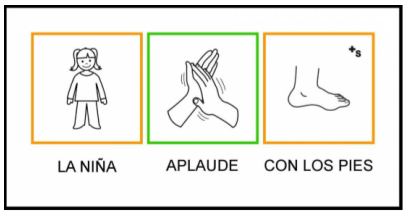






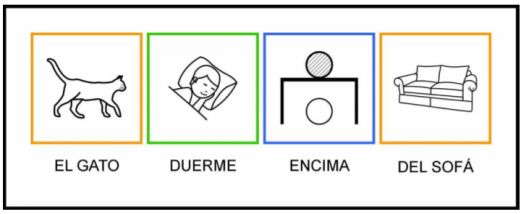

LA NIÑA
SE PONE
UNA CORBATA
EL PAPÁ
BARRE
EL SUELO
CON LA ESCOBA
YO
HAGO PIS
EN LA NEVERA
LOS NIÑOS
PINTAN
UN DIBUJO
EN LA CLASE
LA HIERBA
ES
NARANJA
EL NIÑO
RECORTA
EL PAPEL
CON LA TIJERA

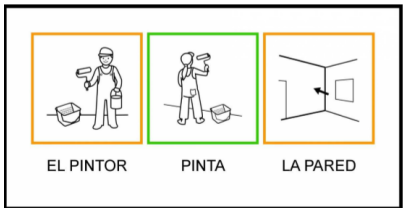

LA NIÑA
SOPLA
LA VELA
EL ABUELO
ECHA
LECHE
EN LA TAZA
LA ZANAHORIA
ES
NARANJA
LA NIÑA
PESCA
UN PEZ
EN EL RÍO
EL NIÑO
SE LAVA
LAS MANOS
EL NIÑO
LANZA
UNA PIEDRA
A LA VENTANA

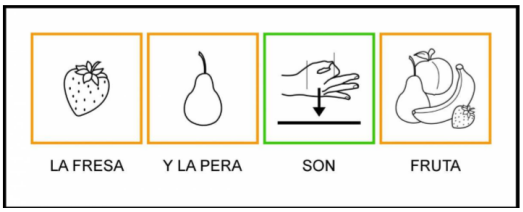

LA NIÑA
APLAUDE
CON LOS PIES

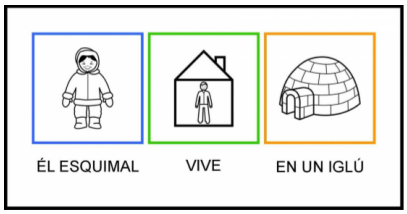

LA NIÑA
SE PONE
ZAPATOS
EN LOS PIES

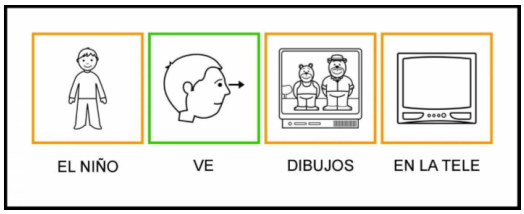

YO
QUIERO
LA PELOTA


EL GATO
DUERME
ENCIMA
DEL SOFÁ

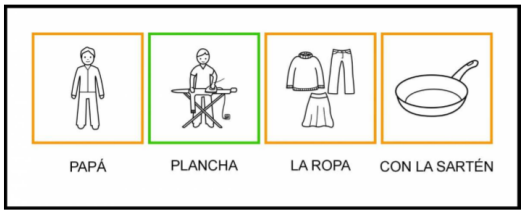

EL PINTOR
PINTA
LA PARED

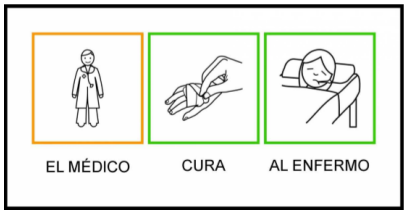

LA FRESA
Y LA PERA
SON
FRUTA


ÉL ESQUIMAL
VIVE
EN UN IGLÚ

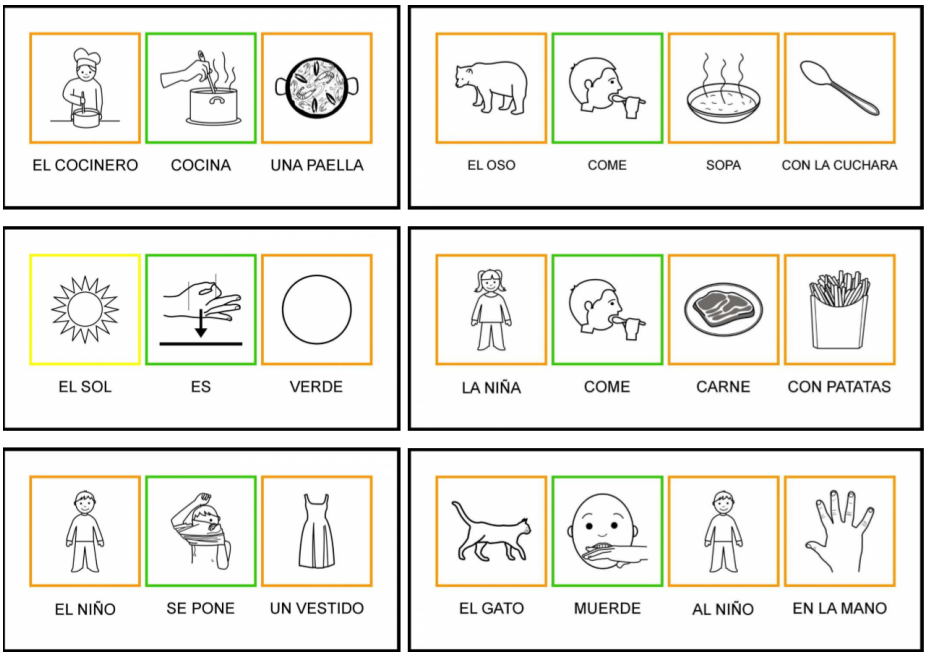

EL NIÑO
VE
DIBUJOS
EN LA TELE

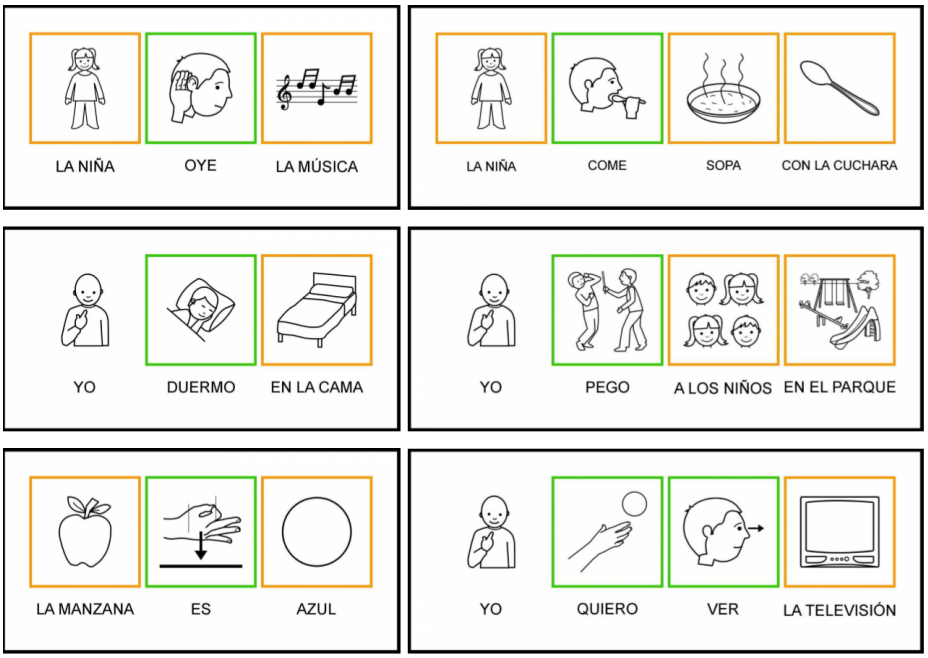

NOSOTRAS
BAILAMOS
BIEN

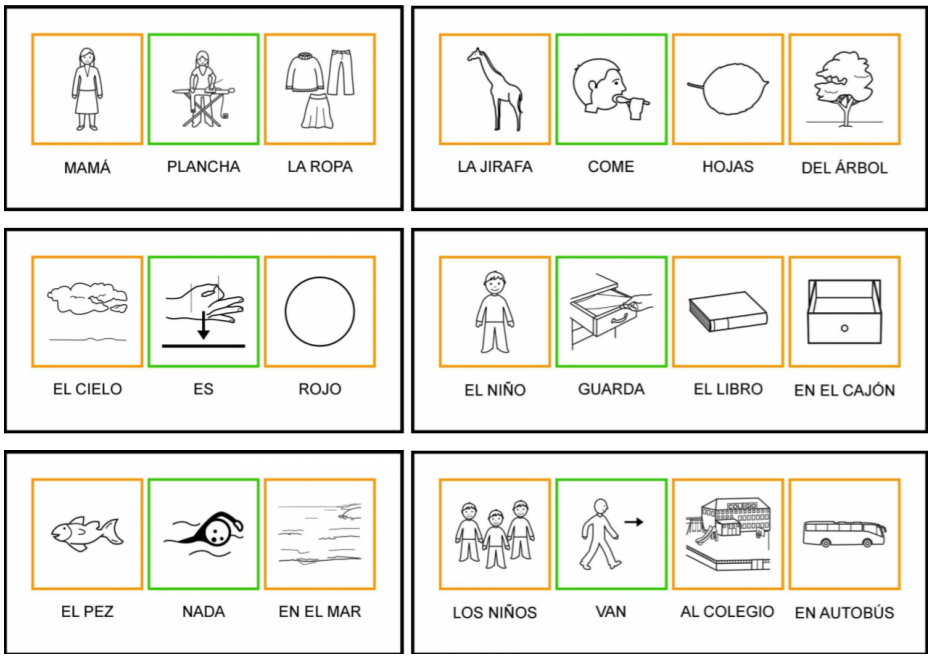

PAPÁ
PLANCHA
LA ROPA
CON LA SARTÉN

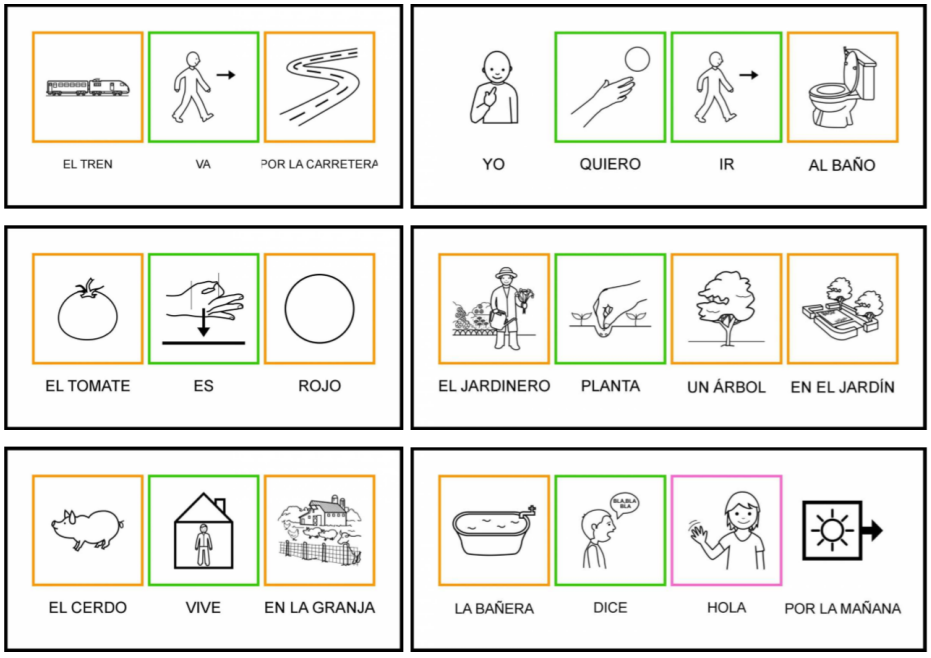

EL MÉDICO
CURA
AL ENFERMO

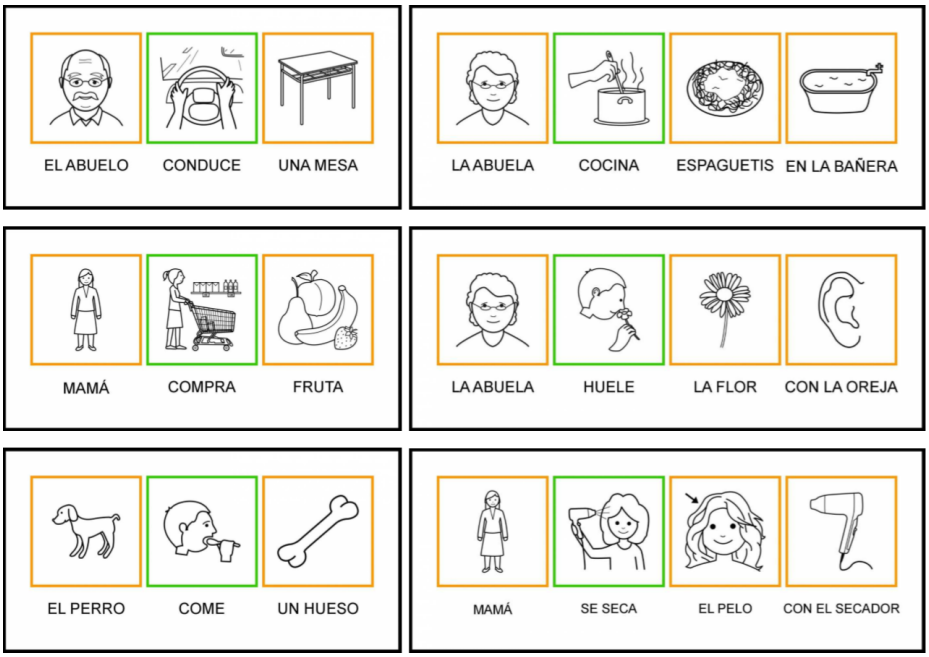

MAMÁ
COSE
UN VESTIDO
AZUL

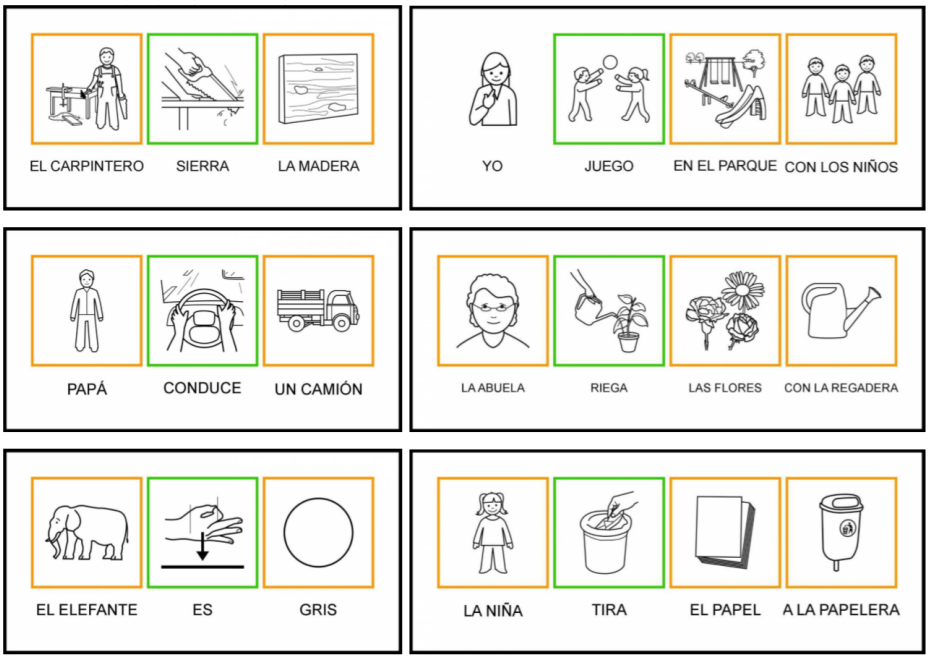

EL COCINERO
COCINA
UNA PAELLA
EL OSO
COME
SOPA
CON LA CUCHARA
EL SOL
ES
VERDE
LA NIÑA
COME
CARNE
CON PATATAS
EL NIÑO
SE PONE
UN VESTIDO
EL GATO
MUERDE
AL NIÑO
EN LA MANO

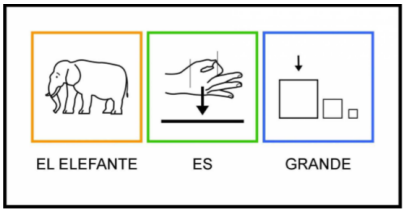

LA NIÑA
OYE
LA MÚSICA
LA NIÑA
COME
SOPA
CON LA CUCHARA
YO
DUERMO
EN LA CAMA
YO
PEGO
A LOS NIÑOS
EN EL PARQUE
LA MANZANA
ES
AZUL
YO
QUIERO
VER
LA TELEVISIÓN

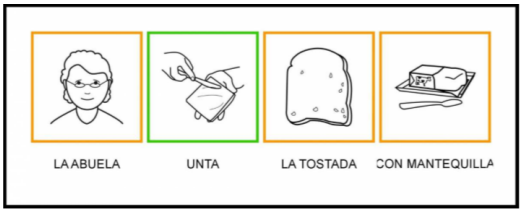

MAMÁ
PLANCHA
LA ROPA
LA JIRAFA
COME
HOJAS
DEL ÁRBOL
EL CIELO
ES
ROJO
EL NIÑO
GUARDA
EL LIBRO
EN EL CAJÓN
EL PEZ
NADA
EN EL MAR
LOS NIÑOS
VAN
AL COLEGIO
EN AUTOBÚS

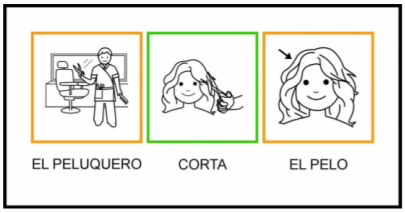

EL TREN
VA
POR LA CARRETERA
YO
QUIERO
IR
AL BAÑO
EL TOMATE
ES
ROJO
EL JARDINERO
PLANTA
UN ÁRBOL
EN EL JARDÍN
EL CERDO
VIVE
EN LA GRANJA
LA BAÑERA
DICE
HOLA
POR LA MAÑANA


EL ABUELO
CONDUCE
UNA MESA
LA ABUELA
COCINA
ESPAGUETIS
EN LA BAÑERA
MAMÁ
COMPRA
FRUTA
LA ABUELA
HUELE
LA FLOR
CON LA OREJA
EL PERRO
COME
UN HUESO
MAMÁ
SE SECA
EL PELO
CON EL SECADOR

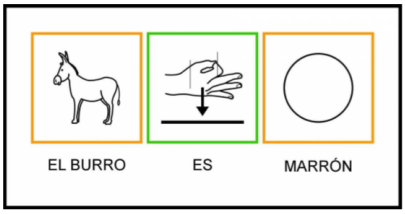

EL CARPINTERO
SIERRA
LA MADERA
YO
JUEGO
EN EL PARQUE
CON LOS NIÑOS
PAPÁ
CONDUCE
UN CAMIÓN
LA ABUELA
RIEGA
LAS FLORES
CON LA REGADERA
EL ELEFANTE
ES
GRIS
LA NIÑA
TIRA
EL PAPEL
A LA PAPELERA

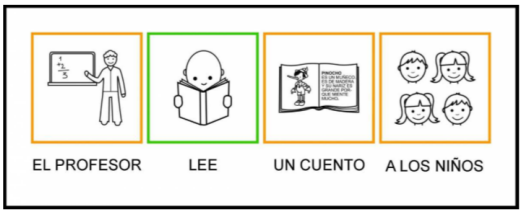

EL ELEFANTE
ES
GRANDE

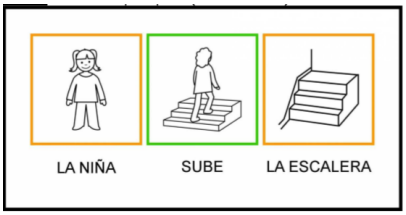

LA ABUELA
UNTA
LA TOSTADA
CON MANTEQUILLA

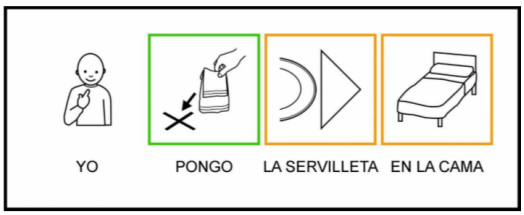

EL PELUQUERO
CORTA
EL PELO


YO
ME BAÑO
CON AGUA
Y CON JABÓN

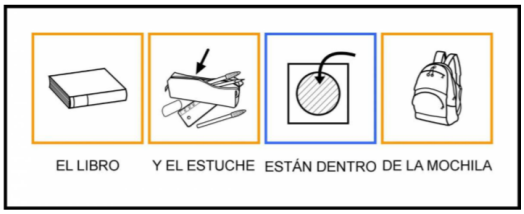

EL BURRO
ES
MARRÓN

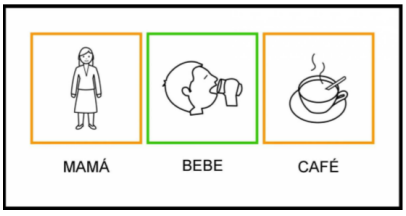

EL PROFESOR
LEE
UN CUENTO
A LOS NIÑOS


LA NIÑA
SUBE
LA ESCALERA

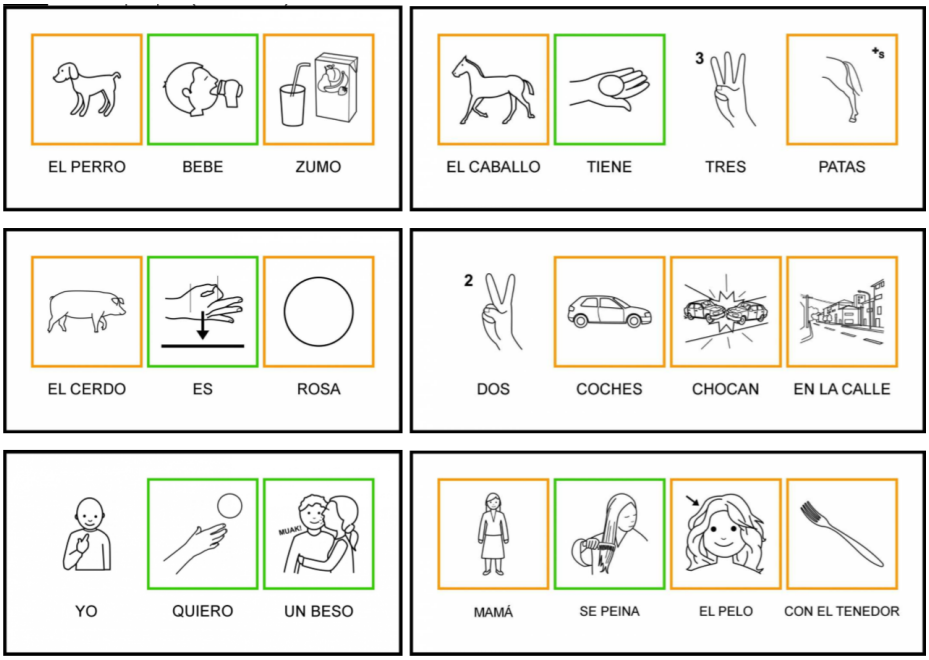

YO
PONGO
LA SERVILLETA
EN LA CAMA

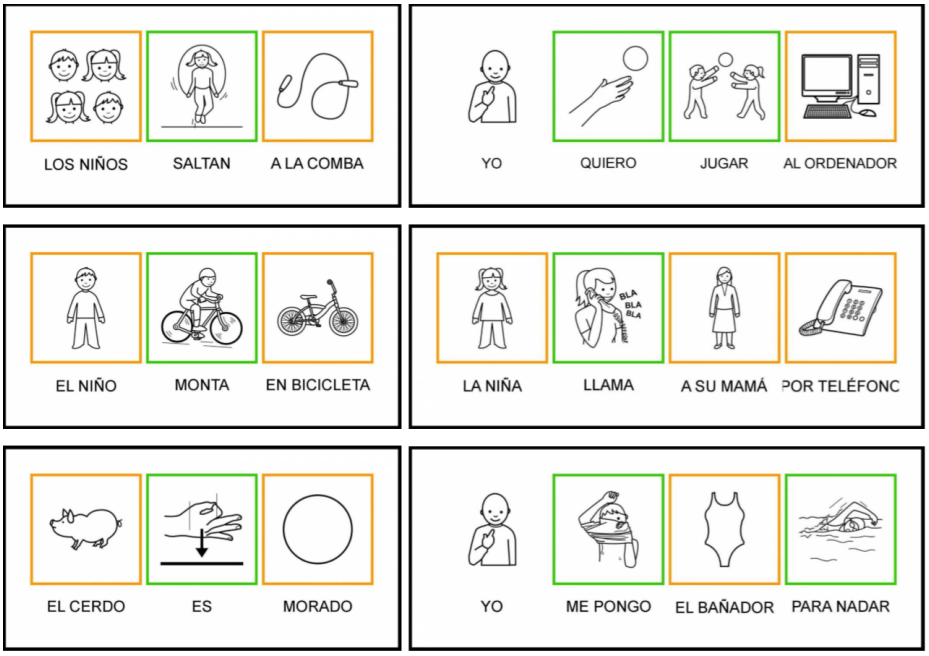

EL MONO
COME
UN PLÁTANO

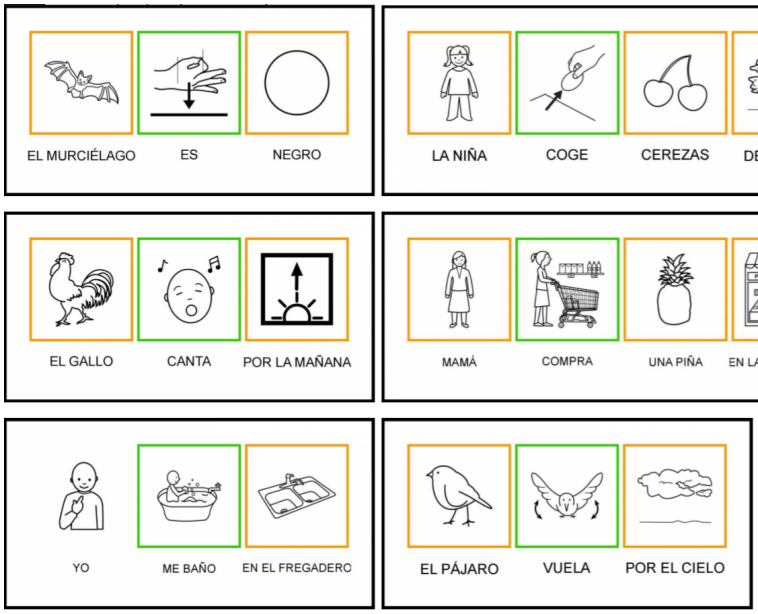

EL LIBRO
Y EL ESTUCHE
ESTÁN DENTRO
DE LA MOCHILA

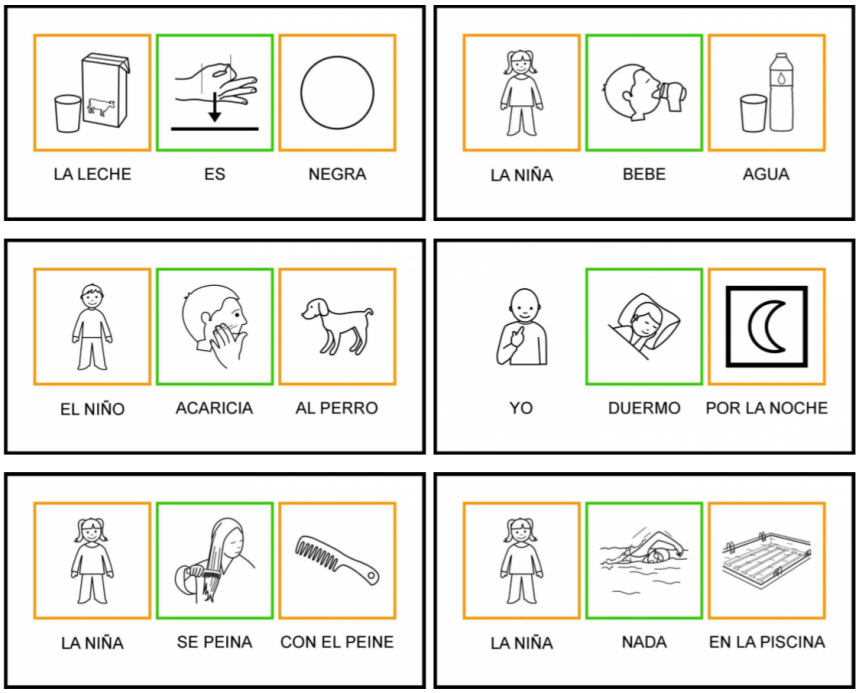

MAMÁ
BEBE
CAFÉ

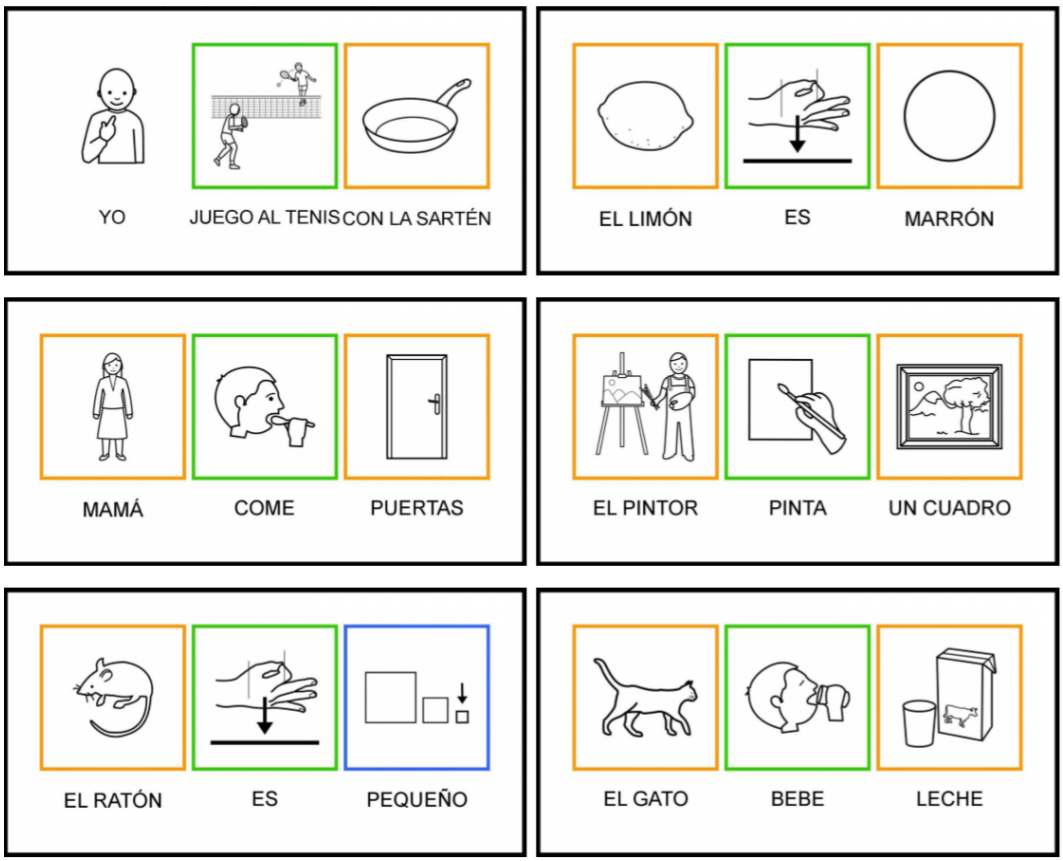

PAPÁ
CUELGA
EL ABRIGO
EN LA PERCHA

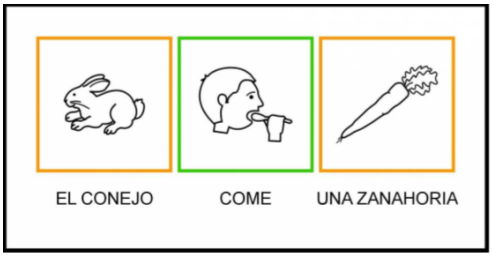

EL PERRO
BEBE
ZUMO
EL CABALLO
TIENE
3
TRES
PATAS
EL CERDO
ES
ROSA
2
DOS
COCHES
CHOCAN
EN LA CALLE
YO
QUIERO
UN BESO
MAMÁ
SE PEINA
EL PELO
CON EL TENEDOR

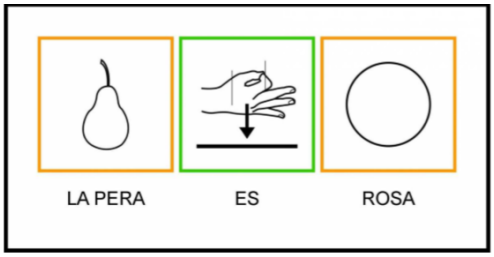

LOS NIÑOS
SALTAN
A LA COMBA
YO
QUIERO
JUGAR
AL ORDENADOR
EL NIÑO
MONTA
EN BICICLETA
LA NIÑA
LLAMA
A SU MAMÁ
POR TELÉFONC
EL CERDO
ES
MORADO
YO
ME PONGO
EL BAÑADOR
PARA NADAR

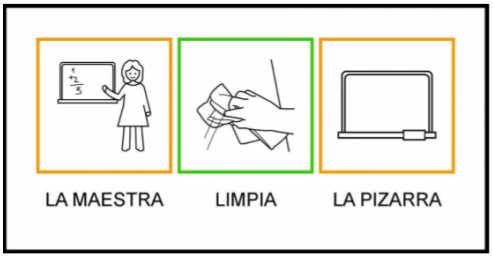

EL MURCIÉLAGO
ES
NEGRO
LA NIÑA
COGE
CEREZAS
DEL ÁRBOL
EL GALLO
CANTA
POR LA MAÑANA
MAMÁ
COMPRA
UNA PIÑA
EN LA PESCADERÍA
YO
ME BAÑO
EN EL FREGADERO
EL PÁJARO
VUELA
POR EL CIELO

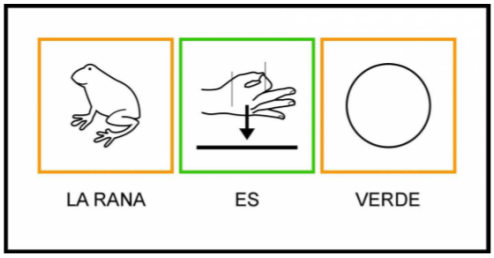

LA LECHE
ES
NEGRA
LA NIÑA
BEBE
AGUA
EL NIÑO
ACARICIA
AL PERRO
YO
DUERMO
POR LA NOCHE
LA NIÑA
SE PEINA
CON EL PEINE
LA NIÑA
NADA
EN LA PISCINA


YO
JUEGO AL TENIS
CON LA SARTÉN
EL LIMÓN
ES
MARRÓN
MAMÁ
COME
PUERTAS
EL PINTOR
PINTA
UN CUADRO
EL RATÓN
ES
PEQUEÑO
EL GATO
BEBE
LECHE

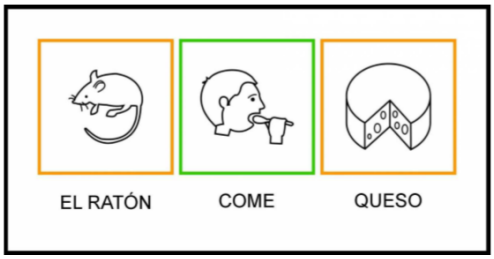

EL CONEJO
COME
UNA ZANAHORIA

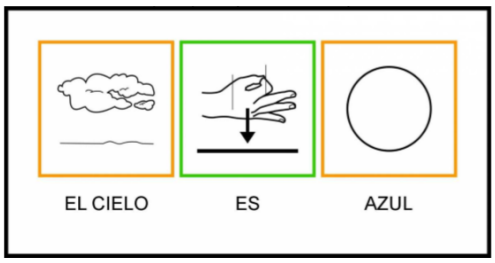

LA PERA
ES
ROSA

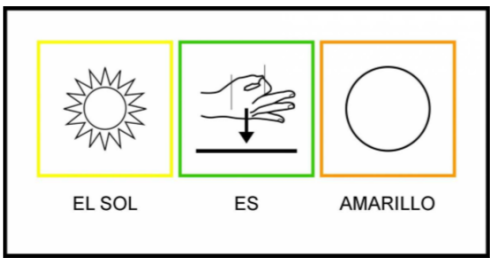

LA MAESTRA
LIMPIA
LA PIZARRA


LA RANA
ES
VERDE


PAPÁ
SE PONE
EL JERSEY


EL RATÓN
COME
QUESO


EL CIELO
ES
AZUL


EL SOL
ES
AMARILLO

Apéndice C

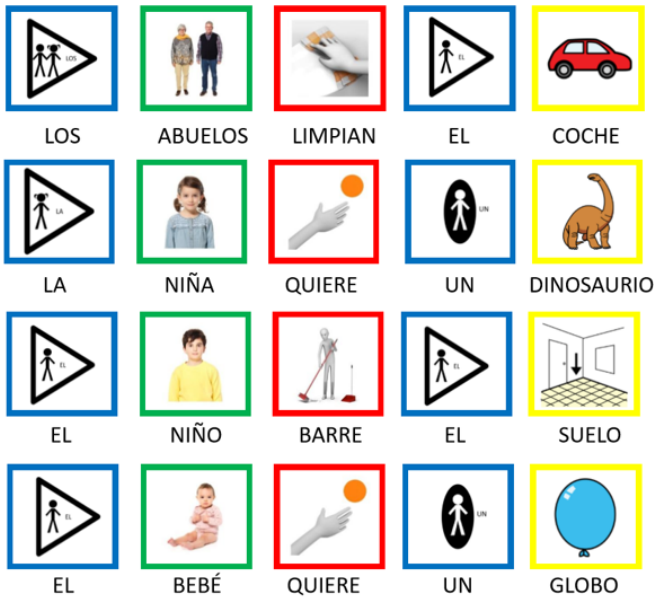
Corpus: conciencia léxica modelos de frases

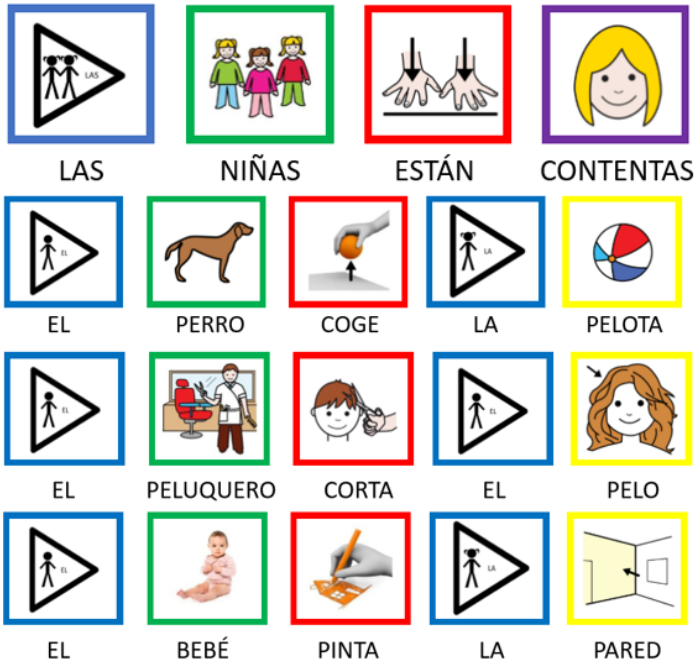

CARLOS
CORTA
CEBOLLA
LA
ABUELA
COMPRA
EL
GATO
HUELE
LOS
ANIMALES
COMEN
LOS
NIÑOS
PINTAN
EL
NIÑO
LEE
LOS
ANIMALES
BEBEN


LA
ABUELA
COME
HELADO
PAPÁ
PLANCHA
LA
ROPA
MAMÁ
GOLPEA
LA
PUERTA
PAPÁ
HACE
UN
BOCADILLO


MAMÁ
LIMPIA
EL
ESPEJO
EL
NIÑO
TIENE
MIEDO
EL
ABUELO
ESTÁ
ENFADADO
EL
NIÑO
ESTÁ
TRISTE

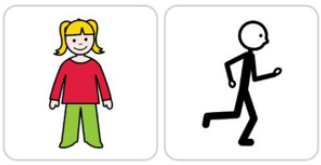

EL
BEBÉ
COME
PAPILLA
Papilla
LOS
NIÑOS
ESCRIBEN
UNA
CARTA
EL
NIÑO
PINTA
LA
PARED
EL
NIÑO
COMPARTE
EL
BOCADILLO


LOS
ABUELOS
LIMPIAN
EL
COCHE
LA
NIÑA
QUIERE
UN
DINOSAURIO
EL
NIÑO
BARRE
EL
SUELO
EL
BEBÉ
QUIERE
UN
GLOBO


LAS
NIÑAS
ESTÁN
CONTENTAS
EL
PERRO
COGE
LA
PELOTA
EL
PELUQUERO
CORTA
EL
PELO
EL
BEBÉ
PINTA
LA
PARED

Apéndice D

Corpus: frases absurdas



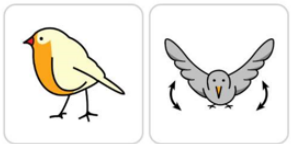
LA NIÑA CORRE

EL AVIÓN RÍE

EL AVIÓN NO RÍE

LA PELOTA CANTA

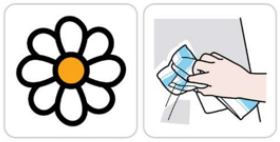
LA PELOTA NO CANTA



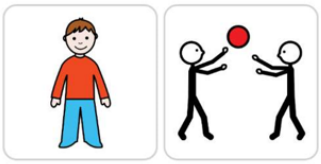
EL PÁJARO VUELA

LA FRESA SE ENFADA

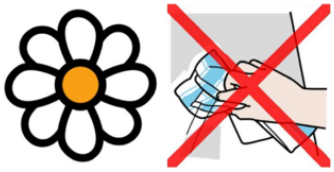
LA FRESA NO SE ENFADA



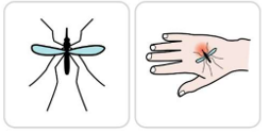
LA FLOR LIMPIA



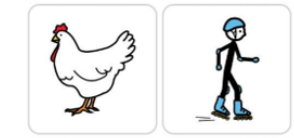
EL NIÑO JUEGA



LA FLOR NO LIMPIA

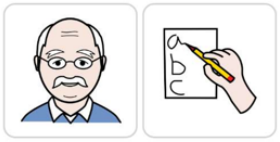


EL MOSQUITO PICA

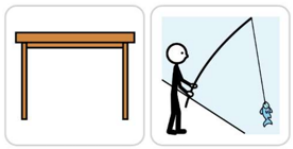


LA GALLINA PATINA

LA GALLINA NO PATINA



EL ABUELO ESCRIBE

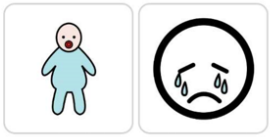


LA MESA PESCA

LA MESA NO PESCA

EL COCODRILO COMPRA

EL COCODRILO NO COMPRA

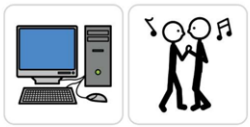


EL BEBÉ LLORA

MAMÁ CONDUCE

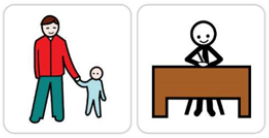
LA SERPIENTE ESTUDIA

LA SERPIENTE NO ESTUDIA



EL ORDENADOR BAILA

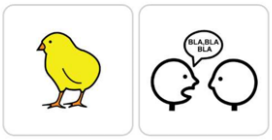
EL ORDENADOR NO BAILA



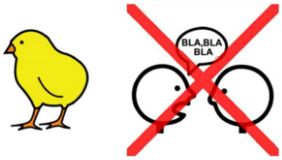
PAPÁ TRABAJA

LA TELE BEBE

LA TELE NO BEBE



EL POLLITO HABLA



EL POLLITO NO HABLA

Bibliografía

Abril Abadín, D., Delgado Santos, C. y Vigara Cerrato, A. *Comunicación Aumentativa y Alternativa*. CEAPAT, 3º edición, 2010.

Álvarez, L. L. *Los sistemas Alternativos y/o Aumentativos de Comunicación: la Comunicación Bimodal como recurso en el aula de Audición y Lenguaje*. Tfg, Universidad de Valladolid, 2013.

Basil, C., Almirall, C. y de la Bellacasa, R. *Comunicación aumentativa: curso sobre sistemas y ayudas técnicas de comunicación no vocal*. Colección Rehabilitación. Instituto Nacional de Servicios Sociales, 1990. ISBN 9788486852566.

Bertola López, E. *Análisis empírico de las caracterásticas formales de los símbolos pictográficos Arasaac*. Tesis Doctoral, 2018.

Correa Piñero, D., Correa Moreno, T. y Pérez-Jorge, D. Comunicació"n aumentativa una introducció"n conceptual y práctica. 2011.

Dale, R., Moisl, H. y Somers, H. *Handbook of Natural Language Processing*. Taylor & Francis, 2000. ISBN 9780824790004.

García Ibáñez, C., Hervás Ballesteros, R. y Gervás, P. Una arquitectura software para el desarrollo de aplicaciones de generación de lenguaje natural. 2004.

Han, J. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. ISBN 1558609016.

Hehner, B. *Símbolos Bliss. Diccionario guía*. Subdirección General de Educación Especial, 1985.

Jozef Trzpis, D. *Adaptación de una herramienta de Generación de Lenguaje Natural al idioma Español*. Phd, Universidad Politécnica de Madrid, 2015.

MARÍN, F. y PÉREZ, F. *Tecnologías de ayuda en personas con trastornos de comunicación*. Colección Logopedia e Intervención Series. Nau Llibres, 2003. ISBN 9788476426821.

MARTIN, R. *Agile Software Development: Principles, Patterns, and Practices*. Always learning. Pearson, 2013. ISBN 9781292025940.

MARTÍN GUERRERO, A. *PICTAR: una herramienta de elaboración de contenido para personas con TEA basada en la traducción de texto a pictogramas*. Phd, Universidad Complutense de Madrid, 2018.

MCDONALD, E. *Sistema Bliss. Enseñanza y uso: enseñanza y uso*. Ministerio de Educación y Ciencia, Subdirección General de Educación Especial, 1985. ISBN 9788436911763.

ROMSKI, M. y SEVCIK, R. Augmentative and alternative communication for children with developmental disabilities. *Mental Retardation and Developmental Disabilities Research Reviews*, vol. 3, páginas 363–368, 1997.

SOCORRO BERNARDOS GALINDO, M. ¿qué es la generación de lenguaje natural? una visión general sobre el proceso de generación. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, vol. 11, páginas 105–128, 2007. ISSN 1137-3601.

URIBE, A., PORTALES, M. y ORTEGA, J. *Apoyo a la comunicación (2018)*. Formación Profesional - Atención a personas en situación de dependencia. Editorial Editex, 2018. ISBN 9788491614234.

VICENTE, M., BARROS, C., PEREGRINO, F. S., AGULLÁ, F. y LLORET, E. La generación de lenguaje natural: Análisis del estado actual. *Computación y Sistemas*, vol. 19, páginas 721 – 756, 2015. ISSN 1405-5546.

WARRICK, A. *Comunicación sin habla*. CEAPAT, 3º edición, 2010. ISBN 0-9684186-0-0.